UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

VISUALIZACIÓN DE DATOS CON MÉTODOS DE REDUCCIÓN DE DIMENSIÓN NO LINEALES

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

MATEMÁTICO

P R E S E N T A

ANDRÉS MEJÍA ZACARÍAS

DIRECTOR DE TESIS

DR. ALESSIO FRANCI

CIUDAD DE MÉXICO, 2021

Agradecimientos

Me gustaría comenzar esta sección agradeciéndome, abrazándome y motivándome para futuras lecturas, pues creo que es el mayor acto de amor propio, sobretodo por los momentos tan difíciles que atravesé durante la escritura de este trabajo.

Agradezco a mi director de tesis, Alessio Franci, que a pesar de los silencios que hubo de mi parte, siempre estuvo presente cuando lo necesité para responder mis dudas. Gracias por aceptar dirigir esta tesis, aún sabiendo que era un tema totalmente nuevo para mi.

Agradezco a mis sinodales, Pedro, Úrsula, Gibrán y Raziel, por sus atinadas observaciones y palabras de aliento sobre mi trabajo.

Agradezco a aquellos profesores de la Facultad de Ciencias que me inspiraron en sus clases, corroborando mi gusto por las matemáticas y reafirmándome que no me equivoqué de carrera.

Agradezco a mi familia nuclear, por su amor incondicional y apoyo incomparable. A mi madre Teté, que me enseñó sobre cómo ser más humano, sobre la ternura y el amor de una madre, pero sobretodo por hacerme la persona que soy. A mi padre José Luis, que siempre ha estado presente en las adversidades, aunque no le tocara. Que desde temprana edad me contagió su amor al conocimiento, a la música y a la creación, su perseverancia y fuerza ante la vida. A mi hermana Lucía, que a pesar de la vida, ha estado presente desde el inicio, con sus risas, su amor y comprensión.

Agradezco a mi mamá Lucía, quien me recuerda constantemente que nos podemos superar, que siempre podemos elegir qué persona queremos ser. Por todo su amor y apoyo, a pesar de las dificultades, y por no darse nunca por vencida. A mi papá Gabriel, que con sus propias limitaciones, me ha enseñado sobre lo real que puede ser el mundo.

Agradezco a mis tíos Andrés y Farid, que desde niño no han dejado de apoyarme ni de creer en mi. Gracias por todo el amor que me han dado y

por estar en aquellos tiempos difíciles, donde no sólo yo necesité de ustedes. A mi tía Elena, que desde pequeños nos cobijó a mi hermana y a mí, junto a sus hijos, y que al día de hoy lo sigue haciendo.

Agradezco a mi tía Cecilia y a mis abuelos, Coco y Farid, por ser esos padres cuando los necesité, por seguir estando presente y atenta de mi.

Agradezco a mis primos, que me han regalado tantas risas y momentos atesorables. En especial a Daniela, que prácticamente es mi hermana mayor, viéndome crecer desde el inicio.

Agradezco a Aarón y Ricardo, que me han enseñado que sí existen otras maneras de relacionarnos. Ambos han sido mi ejemplo a seguir desde que nos conocimos, y quiero que siga siendo así.

Agradezco a Aldo, por hacer de mi vida en la facultad más amena, enseñándome tanto de la vida y de las matemáticas en las largas pláticas que hemos tenido. También, quiero agradecer a la mesa del Prometeo, quienes siempre me alegraban el día con una partida de dominó entre clases, en especial a Bernardo, quien me mostró a perseverar durante la carrera.

Agradezco a las Mondragón Benito, que su cariño me ha demostrado que la familia no necesariamente es de sangre. Gracias por mostrarme cómo ser una persona más fuerte, en especial a Fernanda, mi amiga más antigua, quien hasta el día de hoy no me deja de hacer reír, pero tampoco me deja de escuchar cuando más lo necesito.

Agradezco a Vale, por enseñarme cómo y desde dónde se debe de amar a una persona. Por toda tu escucha y comprensión, desde la ternura y el entendimiento.

Agradezco a Nicolás y a Sara, por compartirme un poco de su amor cada vez que los veo. Por todos aquellos momentos de escucha entre nosotros, pero también de música, que son necesarios para el alma.

Agradezco a Georgette y a todos mis maestros del IE, que es gracias a las bases que me dieron que me he podido desenvolver como lo he hecho en esta vida.

Agradezco a Ale, Diego, Miguel y Emi, por todos los momentos de amistad, imborrables, y por todos los que faltan.

* * *

Este trabajo fue realizado con el apoyo de los proyectos Conacyt Ciencia Básica A1-S-10610 y DGAPA-UNAM PAPIIT IN 102420.

Índice general

Resumen

Este trabajo presenta una comparación entre dos de los métodos más novedosos en el área de reducción de la dimensión en *Machine Learning*: t-SNE y UMAP. Primero se esbozará la noción de qué es un dato y cómo un conjunto de estos tiene una variedad intrínseca, la cual no necesariamente puede ser proyectada linealmente a algún espacio de dimensión menor. Exploraremos estas razones para justificar la búsqueda de métodos de reducción de dimensión no lineales e intentaremos comprender la teoría detrás de la existencia de dichas proyecciones.

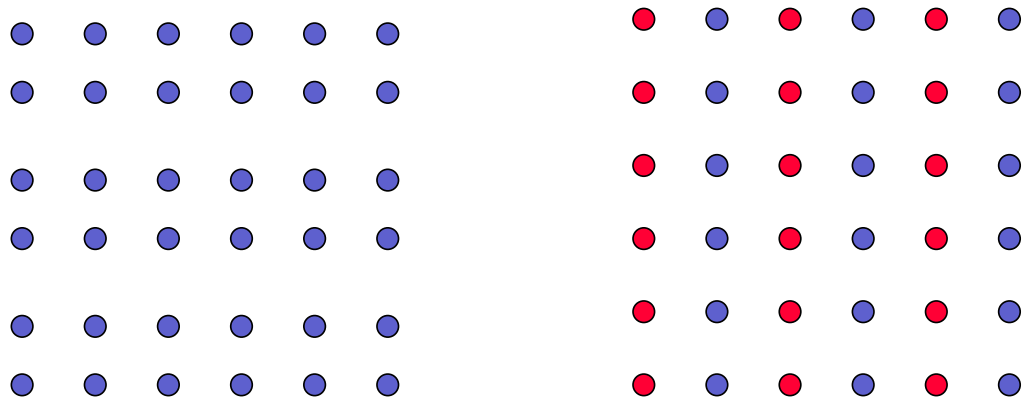
Posteriormente hablaremos de cada algoritmo por separado, estudiando la teoría matemática detrás de ellos y cómo es que repercutirá en los resultados obtenidos, desde comprender cómo es que las funciones de costo afectarán el desempeño temporal de los métodos, hasta ver cómo es que afectarán en la preservación de la morfología de las estructuras estudiadas.

Capítulo 1

Introducción

Los humanos somos seres que basamos nuestra vida en la mirada. Nacemos con la predisposición a identificar las caras de las personas cercanas a nosotros, incluso antes de diferenciar objetos y colores, y mucho antes de entender los sonidos a nuestro alrededor y de comenzar a balbucear queriendo imitar el lenguaje [1], [2]. Históricamente la visión ha fungido un papel importante para la supervivencia, ya que permite analizar nuestro alrededor y estar alerta en caso de amenazas, para encontrar comida o refugio. A través del tiempo este sentido se fue afinando, a tal punto, que gran parte de nuestra actividad cerebral consiste en procesar lo que visualizamos cada segundo. Estas necesidades provocaron que el cerebro (no solamente humano) desarrollara una gran habilidad para crear y distinguir patrones en los objetos que hay a nuestro alrededor, agrupar por sus similitudes morfológicas o por su posición en el espacio, así como vemos en la figura 1.1.

La ciencia de datos ha tenido un crecimiento importante en los últimos años debido a que podemos recolectar una gran cantidad de datos de las interacciones de usuarios a lo largo del internet, principalmente en redes sociales. Como usuarios vertemos información en ellas, haciéndolo activamente al publicar o subir cualquier tipo de contenido, o pasivamente, viendo contenido de otros usuarios sin ningún tipo de interacción adicional. Toda esta información se recolecta para posteriormente limpiarla, analizarla y encontrar patrones en la información. Este procedimiento es conocido como minería de datos [3]. Las grandes empresas de tecnología, al ofrecer productos gratuitos, tienen que conseguir de algún lado sus recursos, pues si bien estas empresas ofrecen plataformas de interacción por internet, en realidad son compañías que se dedican a la publicidad, por lo que descubrir patrones en los datos recaba-



(a) (b)

Figura 1.1: En (a) vemos un ejemplo donde agrupamos por la posición de los puntos. En (b) por las similitudes.

dos de sus usuarios les ayuda a generar predicciones en su comportamiento y así generar publicidad personalizada para maximizar la probabilidad que el usuario sea enganchado por algún producto. Podemos recordar el caso de *Cambridge Analytica*, donde dicha empresa, obteniendo información de alrededor de 50 millones de usuarios de Facebook, lograron generar tendencias y modificaron la intención de voto en las elecciones de Estados Unidos del 2016 [4]. Para que este proceso sea cada vez más exitoso, se deben de desarrollar herramientas que ayuden a procesar dicha información de manera masiva.

Considerando que los usuarios de redes sociales generan información constantemente (que también debe ser contemplada para obtener resultados más precisos), se deben de usar algoritmos que actualicen sus parámetros frecuentemente. La inteligencia artificial (IA) en conjunto con ciencia de datos, se volverán una herramienta primordial a la hora de minar información y entrenar modelos. Por ejemplo, se han podido entrenar algoritmos que logran predecir cáncer pulmonar en etapas tempranas, superando incluso diagnósitcos realizados por oncólogos veteranos. Dicho modelo fue entrenado con 40,000 tomografías computarizadas y puede generar diagnósticos con un 94 % de certeza.

Dentro de la IA, podemos encontrar que *machine learning* (ML) es un subconjunto de ella, que en pocas palabras, es el conjunto de técnicas cuya efectividad mejorará con el tiempo gracias a que serán entrenadas. Dos de los

acercamientos más importantes para resolver problemas son el aprendizaje supervisado y el no supervisado. Por un lado el aprendizaje supervisado se define por usar información etiquetada para supervisar que dichos algoritmos estén modelando de manera adecuada la información que reciben (*input*), y así afinar los parámetros para aumentar la exactitud con la que predicen los resultados (*output*). El aprendizaje no supervisado, que es en el cual nos enfocaremos en esta tesis, son algoritmos que no dependen de la presencia de un *output* para trabajar, si no que se enfocan en inferir información de nuestro conjunto de información dado [5]. Así, estas técnicas ayudan a resolver problemas donde queremos encontrar patrones en nuestros datos, por ejemplo: agrupar datos no etiquetados por algún tipo de similitud o encontrar relaciones entre las variables que estemos midiendo de nuestros datos. Otro ejemplo de problema donde se usa este tipo de aprendizaje es la reducción de dimensión.

Entre más sepamos de un objeto de estudio, mejor lo conocemos. Regresando al ejemplo de las redes sociales, se pueden medir distintos campos de la interacción entre los usuarios: desde la edad y el tipo de contenido que consume, hasta medir el tiempo que tardan viendo cada publicación. Cada instancia que se mide se puede representar como una entrada del dato que representa a ese usuario dentro de una gigantesca base de información. Dichos datos pueden llegar a tener miles de dimensiones, complicando su análisis debido a la capacidad computacional que se necesitaría para procesarlos, pero también porque somos incapaces de poder visualizar ese conjunto de datos. Es por eso que dentro del área de la reducción de dimensión se estudian métodos para poder disminuir el número de dimensiones a 2 ó 3, afectando lo menos posible las estructuras geométricas globales y locales de los datos, para así poder hacer un estudio de estos y encontrar algún tipo de patrón o relación en la información. Hacer un tratamiento previo de la información con alguno de estos métodos permite trabajar con mayor agilidad, consume menos recursos computacionales y tiempo. Por último, pero no menos importante, permitirá que los analistas tengan una mejor percepción del acomodo de la información en el espacio para comprender mejor su geometría.

La vista evolucionó para que como especie pudiéramos sobrevivir, pero estos mecanismos han impactado de múltiples maneras nuestro desarrollo como individuos, y en particular, al desarrollo de conocimiento. Es así que la visión también ha fungido un papel importante en la creación del conocimiento, pues la posibilidad de ver plasmada en algún formato las abstracciones que habitan en nuestro imaginario, nos permite poder manipularlo y estudiarlo

desde perspectivas que no teníamos previamente.

Este trabajo de tesis tiene por intención hacer una comparación de los dos métodos de reducción de dimensión más usados actualmente: t-SNE (*t-Distributed Stochastic Neighbor Embedding*) y UMAP (*Uniform Manifold Approximation and Projection for Dimension Reduction*). Se busca usar un par de ejemplos juguete para estudiar qué tan bien preservan las estructuras locales y globales, pero también qué tan bien logran clasificar un conjunto de datos etiquetado. De igual forma se busca que esta tesis sea un proyecto autocontenido y la forma en la que fue redactada es para que otros alumnos de nivel licenciatura puedan tener un texto introductorio a la reducción de dimensión, y por otro lado, que los lectores que no tienen ningún tipo de antecedente en matemáticas o alguna disciplina similar puedan entender (aunque sea lo mínimo) de este tema.

Capítulo 2

Un vistazo a la reducción de dimensión de datos mediante la hipótesis de la variedad

2.1. El encaje suave

Si en los primeros semestres de la carrera me hubieran preguntado cómo poder reducir de dimensión un conjunto de puntos que vive en un espacio alto-dimensional a otro con menos, hubiera contestado que usando la proyección canónica y ya, no habría mayor problema. En teoría debería ser así de sencillo, pero al investigar un poco más nos damos cuenta que hay varios detalles que no son nada fáciles de resolver.

Primero explicaré lo que es una proyección en el campo del álgebra lineal: imaginemos que tenemos una esfera suspendida por un hilo. Si alumbráramos con una linterna a la esfera de manera perpendicular a la pared más cercana, la sombra tendría forma de un disco (un círculo relleno); la proyección fue la acción de mandar la esfera a la pared (un plano bidimensional). Ahora, ¿qué pasaría si proyectamos la esfera a un espacio de una dimensión? El único espacio que tiene una dimensión, sin complicarnos más, es una recta que se extiende infinitamente por ambos extremos, como vemos en la figura 2.1. Entonces, el resultado de esa proyección sería un segmento de esa recta cuyo largo es el diámetro de la esfera. En resumen, una proyección es la acción de mandar un objeto de un espacio a otro que sea de menor dimensión.

Ahora, uno de los problemas que surgen es que existen ciertos conjuntos

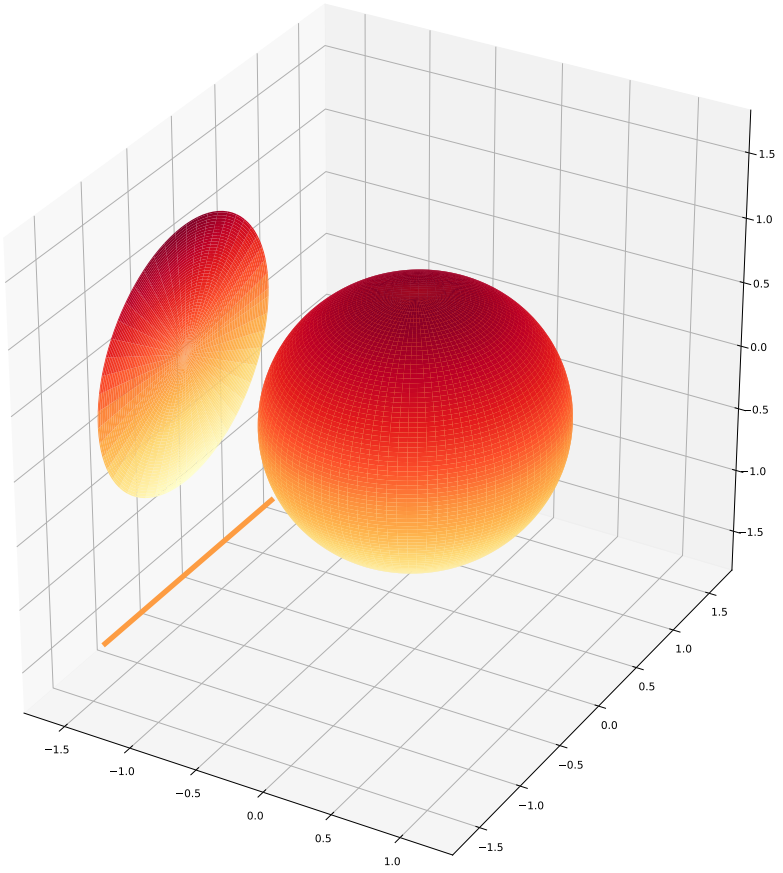


Figura 2.1: Proyección de una esfera.

que no pueden ser proyectados mediante transformaciones lineales, como el "Swiss Roll", que es un conjunto definido como la imagen de la siguiente función:

$$f(\theta, \rho) = (\theta\cos(\theta), \rho, \theta\sin(\theta)), (\theta, \rho) \in [0, 2\pi n] \times [0, 1], n \in \mathbb{N} \tag{2.1}$$

(n será el número de vueltas que tenga). A partir de este momento se denotará como el conjunto S.

Las funciones lineales actúan de maneras muy específicas sobre los conjuntos en las que están definidas, estas pueden estirarlos o contraerlos, cambiar

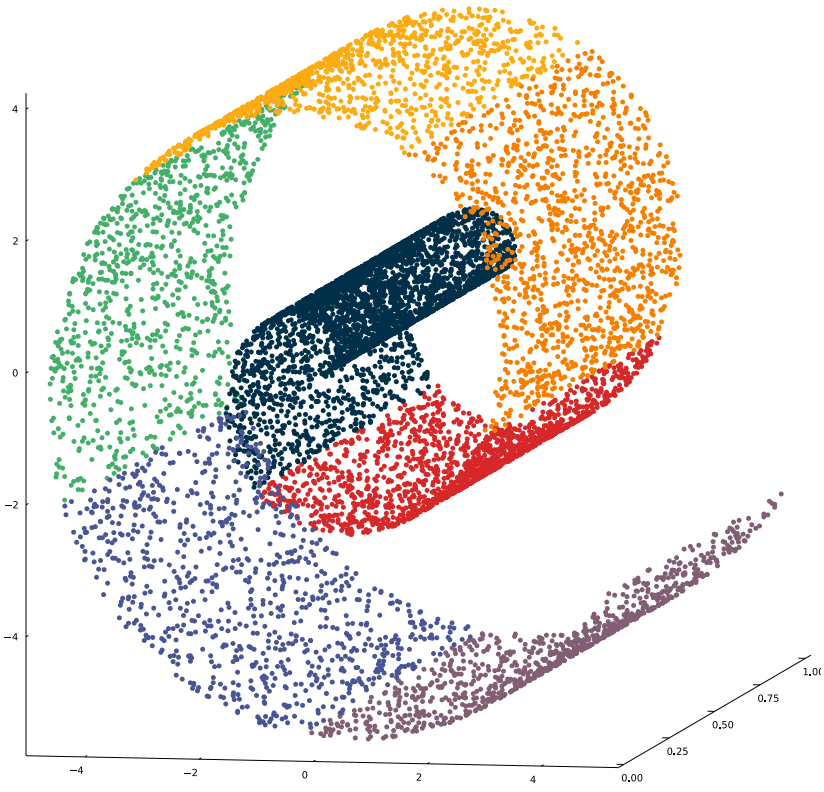


Figura 2.2: Swiss Roll, gráfica de la función 2.1

el ángulo entre los ejes coordenados e incluso rotarlos o reflejarlos sobre alguna recta. En la figura 2.3 una transformación lineal está actuando en la función seno.

Si se quisiera visualizar al conjunto S en el plano sin aplastarlo (encimar puntos del mismo conjunto), nuestra intuición nos diría que lo desenrollemos, ¿pero es esto posible usando funciones lineales?

Proposición 1. *No existe ninguna transformación lineal* $T : \mathbb{R}^3 \mapsto \mathbb{R}^2$ *tal que* $T^{-1}(T(S)) = S$.

Demostración. Sea T una transformación lineal de $\mathbb{R}^3$ a $\mathbb{R}^2$, sin pérdida de generalidad esta es de la forma $T(x, y, z) = (x', y')$. Sea l una recta tal que $|l \cap S| \geq 2$ y además $l \cdot n = 0$ donde $n = \alpha x + \beta y$, con $\alpha, \beta \in \mathbb{R}$, i.e. n vive en el plano xy. Hay que notar que dicha recta existe porque S da al menos dos

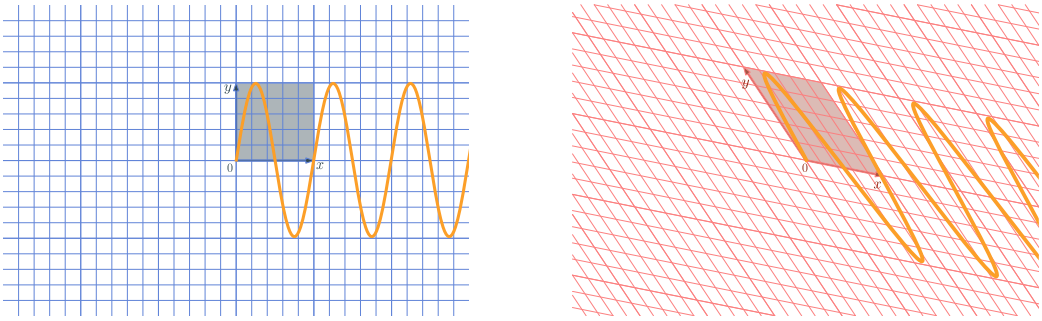


Figura 2.3: La función seno antes y después de una transformación lineal.

vueltas en sí mismo. Sean $\xi_1 = (x, y, z_1)$ y $\xi_2 = (x, y, z_2)$ en la intersección, entonces:

$$T(\xi_1) = T(\xi_2) = (x', y') \implies T(\xi_1) - T(\xi_2) = 0$$

Como la inversa de una transformación lineal es lineal:

$$T^{-1}\left(T(\xi_1) - T(\xi_2)\right) = 0 \implies T^{-1}\left(T(\xi_1)\right) = T^{-1}\left(T(\xi_2)\right) \implies \xi_1 = \xi_2$$

Lo cual es una contradicción.

■

La proposición anterior justifica por qué las funciones lineales no nos funcionan para proyectar este tipo de conjuntos. Un ejemplo podría ayudar a entender la demostración anterior: si pensamos en un árbol de navidad e hiciéramos una lista con las coordenadas de cada uno de los adornos para después visualizarlos en el plano, se perdería la información sobre la altura de dichos adornos.

En el proceso de diseñar un experimento se tienen que definir las formas en las que se capturarán los datos, pero de igual manera los métodos de visualización que se emplearán, ya que de esto depende el tipo de resultados que esperamos obtener. Al proyectar, lo que hacemos es eliminar coordenadas del vector, generando pérdidas de información. Si proyectáramos el conjunto S con una transformación lineal, se aplastaría en el plano, lo cual contradiría la inyectividad que estamos buscando. Es por esto, que al no encontrar una solución satisfactoria a este problema, habrá que intentar con otro tipo

de transformaciones: las no lineales. A continuación se describirán algunos conceptos que ayudarán a generalizar las ideas planteadas hasta ahora:

Definición 2.1.1. Sea X un espacio topológico. Decimos que X es **II-numerable** (segundo numerable) si existe una base numerable para la topología de X.

Definición 2.1.2. Sea X un espacio topológico. Decimos que X es un espacio **Hausdorff** si para cualquier par de puntos x, y distintos, existen dos vecindades U y V de X y Y respectivamente, tales que $U \cap V = \varnothing$.

Definición 2.1.3. Sea X un espacio topológico. Decimos que X es un espacio **localmente Euclideano** n-dimensional si $\forall\, x \in X$ existe U abierto tal que hay un homeomorfismo $\phi : U \to V$, donde $V \subset \mathbb{R}^n$ es un abierto.

Definición 2.1.4. Una **variedad topológica** n-dimensional M es un espacio topológico Hausdorff, II-numerable y localmente Euclideano.

Definición 2.1.5. Sean X y Y espacios topológicos. Una función continua e inyectiva $f: X \to Y$ es un **encaje topológico** si $f: X \to f(X)$ es un homeomorfismo.

En pocas palabras se dice que una variedad es un espacio que localmente se comporta como $\mathbb{R}^n$, es decir, que trabajar con puntos cercanos con la misma practicidad que en los espacios euclideanos. Supongamos que tenemos una variedad M n-dimensional que no se puede proyectar en el plano mediante una transformación lineal. La idea detrás de los algoritmos que motivan esta tesis será la siguiente: encontraremos una función no lineal entre $\mathbb{R}^2$ y M, viendo que esta es, realmente, un homeomorfismo. Así, existirá la función inversa que irá de M en $\mathbb{R}^2$, "proyectando" la variedad en el plano[1][6]. Pero, ¿es esto del todo cierto?

Proposición 2. *Sea B una bola cerrada de $\mathbb{R}^2$ y M una n-variedad. Sea $\psi : B \to M$ una función continua e inyectiva. Entonces $\psi : B \to \psi(B)$ es un homeomorfismo.*

Demostración. Se puede ver que ψ es suprayectiva con respecto a su imagen, de ahí la biyectividad. Ahora, B es cerrada y acotada, por teorema de Heine-Borel es compacta, entonces $\psi(B)$ es compacto (pues la compacidad

[1]Notemos que podríamos proyectarlo a un subconjunto del plano siempre y cuando este sea cerrado y acotado.

es un invariante topológico). Como M es Hausdorff, entonces $\psi(B)$ es cerrado. Como ψ es una función cerrada, continua y biyectiva, entonces es un homeomorfismo. ■

Demostrado esto, vemos que ψ es un encaje topológico de $\mathbb{R}^2$ en M. Como ψ restringida a su imagen es un homeomorfismo, podemos definir su función inversa como $\phi : \psi(M) \to B$. Esta función es la proyección de la que hemos estado hablando, pues es una función de nuestro conjunto alto-dimensional al plano.

Notemos que el dominio de ϕ es solo un subconjunto de M, proyectando nada más una parte de la variedad y perdiendo el resto. Cuando proyectamos a S en el plano esto no ocurre, debido a que su dimensión es 2, la misma que el plano. Si bien cada punto de S es una combinación lineal de los vectores canónicos de $\mathbb{R}^3$, decimos que su dimensión es 2 pues existe un homeomorfismo entre $\mathbb{R}^2$ y S. La dimensión es un invariante topológico, por lo que para variedades cuya dimensión es mayor a 2, no existirá un homeomorfismo que proyecte a la variedad en el plano. Recordemos que el objetivo es poder reducir de dimensión a todo el conjunto, entonces nos enfrentamos a otra problemática debido a que matemáticamente hay muchas restricciones ante el razonamiento que hemos estado desarrollando, pues nos enfrentamos a problemas que no son triviales. Sabiendo que se perderá una parte de la variedad a la hora de proyectar, habrá que minimizar esta pérdida.

2.2. El encaje discreto

En la vida real no existen los infinitos, todo lo que conocemos es finito o tiene un fin, ya sea un viaje de la Tierra al Sol o incluso la cantidad de átomos en el universo. Una computadora procesa una cantidad finita de información dada una unidad de tiempo, por lo que a la hora de analizar datos, también serán una cantidad finita. Pero esto a las matemáticas nunca les ha importado mucho; durante nuestra formación como matemáticos se nos enseña a trabajar con infinitos, incluso unos más abundantes que otros, pero al momento de querer aplicar lo que sabemos al respecto tenemos poco éxito. En el capítulo anterior mostré la no trivialidad de proyectar variedades en el plano, pero ahora veremos que este problema se facilita al trabajar con una cantidad finita de datos.

Prácticamente todos hemos interactuado con imágenes digitales y seguramente se ha oído hablar que estas tienen una determinada resolución, lo cual

en pocas palabras significa la densidad de pixeles que esta tiene. El número de pixeles es una unidad de medida para saber qué tan ancha o alta es una imagen, es decir, se dice que una imagen mide x pixeles de ancho por y pixeles de alto. Ahora supongamos que tenemos un conjunto $A = \{a_1, ..., a_l\}$ de imágenes que miden 28 pixeles de ancho por 28 pixeles de alto, teniendo en total 784 pixeles por imagen. A cada pixel le podemos agregar un valor entre el 0 y 1, donde 0 significa que el pixel es negro, 1 significa que el pixel es blanco y cualquier valor en ese intervalo representa un tono de gris. Pensando en que cada imagen consiste de 784 pixeles, donde cada pixel tiene un valor en $[0, 1]$, entonces cada imagen se puede interpretar como un vector con 784 entradas, por lo que el conjunto de todas las imágenes existentes en blanco y negro[2] de 28×28 pixeles, viven en el cubo

$$C = \prod_{i=1}^{784} [0, 1]^i \subset \mathbb{R}^{784}.$$

Ahora seleccionemos todas las imágenes que viven en este espacio que sean de un reloj donde en cada foto marque una hora distinta. Si por ahora consideramos que el reloj no tiene el segundero, entonces en total habrían $60 \times 12 = 720$ fotos. Como cada una de estas representa un vector dentro de C , al cambiar estas fotos una por una, podríamos ver cómo estos vectores se moverían de forma continua, pues el único cambio de una foto de un minuto a otro es la posición del minutero, por lo que el vector a_i se moverá relativamente poco a la posición a_{i+1}. Si a este análisis agregáramos las fotos que incluyen el segundero, habrían 60 veces más imágenes, por lo que el cambio de una foto a otra sería mucho más suave. Esta curva, o incluso una variedad de dimensión mayor, será una subvariedad de C. Este concepto es conocido como la hipótesis de la variedad, la cual ha servido en diversos campos de Machine Learning, pues saber qué datos de la vida real vivirán en alguna variedad de dimensión D, con $D \leq n$ ayudará a tener que trabajar con menos dimensiones, lo cual tendrá un ahorro de tiempo y computacional importante.

Consideremos un conjunto $X = \{x_1, ..., x_l\} \subset \mathbb{R}^n$ de datos, podemos suponer que estos pertenecen a una variedad de dimensión D encajada en $\mathbb{R}^n$. Recordando la sección anterior, podemos suponer que existe un encaje ψ para después encontrar su inversa y que esta funja como proyección, así como

[2]Notemos que la mayoría de las imágenes que vivirán en el cubo serán "ruido".

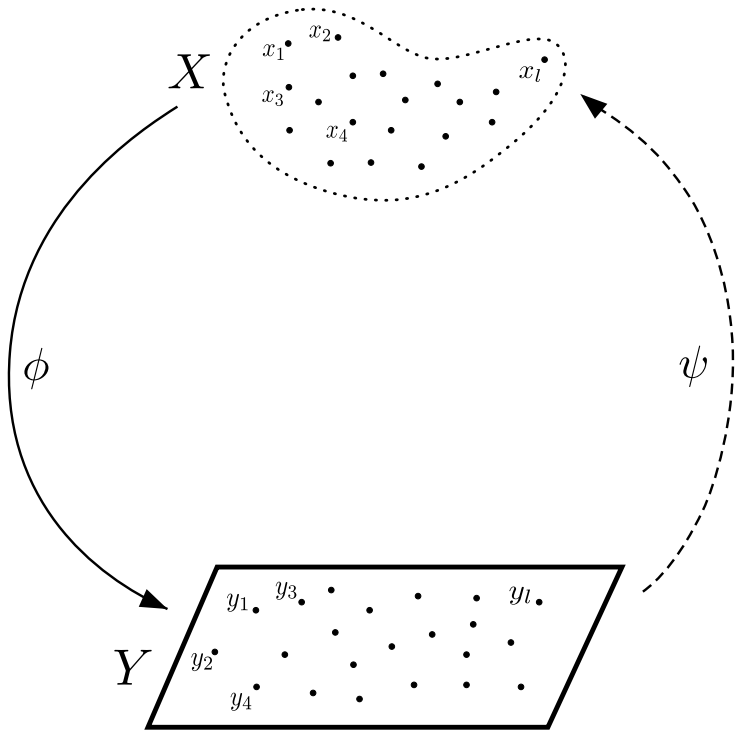


Figura 2.4: Reducción de dimensión de un conjunto discreto.

observamos en la figura 2.4. A pesar de estar trabajando con un conjunto discreto, la dificultad de encontrar una expresión analítica de la proyección sigue siendo alta. Por esta razón generaremos un conjunto $Y = \{y_1, ..., y_l\} \subset \mathbb{R}^2$ de manera aleatoria y definiremos $\psi : Y \longrightarrow X$ que mandará a $y_i \mapsto x_i$. Las funciones entre conjuntos discretos cumplen que son homeomorfismos, por lo que existe su función inversa $\phi : X \longrightarrow Y$ tal que $\phi = \psi^{-1}$.

Al final de la sección anterior se discutió cómo es que la dimensión topológica de M cumple un papel importante para saber si esta podía ser proyectada al plano. Al incrementar su dimensión, requeriremos más tiempo para generar una reducción de dimensión apropiada, pues significa que la cantidad de propiedades con las que trabajamos aumentarán, y por lo tanto, también la información que procesaremos. Los datos de X, al estar sobre una variedad M tendrán cierta estructura topológica que se preservaría en

su mayoría si las proyecciones fueran lo suficientemente buenas. ¿Pero cómo se construye un buen encaje? Si bien no podemos saber exactamente cómo es el acomodo de los elementos de X, se puede saber qué tan densas son ciertas regiones, lo cual nos permitirá intuir cómo es la estructura topológica de M. Una propiedad que querríamos tomar en cuenta para construir a ϕ es que preserve la distribución de los puntos de X en Y. El primer paso a tomar en esta dirección será describir de manera probabilística cómo están situados los puntos en M, por lo que construiremos una distribución de probabilidad por cada punto de X.

Capítulo 3

t-SNE y UMAP

3.1. SNE

SNE (*Stochastin Neighbor Embedding*) es un algoritmo que fue desarrollado por *Geoffrey Hinton* [7] en 2003. Este capítulo comenzará explicando la idea central, para así comprender las mejoras que se realizaron y que dieron lugar a t-SNE (*t-Distributed Stochastin Neighbor Embedding*).

Lo primero que hará SNE es dejar de trabajar con las nociones de distancia euclidiana[3] y comenzará a usar probabilidades. Nombraremos al conjunto de datos que queremos reducir de dimensión como $X = \{x_1, x_2, ..., x_p\} \subset \mathbb{R}^n$. La similitud entre los datos x_i y x_j estará dada por una distribución Gaussiana con respecto a su distancia euclidiana. Después, normalizaremos estas similitudes convirtiéndolas en una distribución condicional de probabilidad $p_{j|i}$[4], que se interpreta como la posibilidad de que x_i pueda elegir como vecino a x_j, sobre la posibilidad de que elija a todos los demás. La expresión analítica estará dada de la siguiente manera:

$$p_{j|i} = \frac{\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)} \quad (3.1)$$

donde σ_i^2 es la varianza, la cual es una medida de dispersión de los datos con

[3] Si bien el concepto de distancia euclidiana se usará en las normas de las densidades Gaussianas, lo que se pierde es el concepto de linealidad en la cercanía de los puntos.

[4] $p_{j|i}$ no se refiere a una probabilidad condicional, es mera notación.

respecto a la media, que en este caso es el valor x_i. Si damos una interpretación geométrica, podríamos pensar que existe una bola abierta alrededor de x_i con radio σ_i^2, donde si un punto x_j está dentro de dicha bola, entonces se considerará vecino de x_i con una probabilidad $p_{j|i}$.

Para el espacio bajo dimensional, crearemos un conjunto $Y = \{y_1, y_2, ..., y_n\}$ de manera aleatoria, el cual consideraremos como la imagen en el espacio de baja dimensión de los puntos de X. En visualización de datos usualmente se pide que $Y \subset \mathbb{R}^2$, pues es más fácil poder interpretar y diferenciar los distintos grupos de puntos cercanos entre sí (o *clusters*). De igual manera que en el caso alto-dimensional, calcularemos la probabilidad de que y_i sea vecino de y_j, solo que en este caso fijaremos $\sigma_i = \frac{1}{\sqrt{2}}$. La expresión matemática será similar al caso anterior:

$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)} \tag{3.2}$$

Notemos que como nos interesa saber si dos puntos son vecinos, no tiene sentido calcular $p_{i|i}$ ni $q_{i|i}$, por lo que los definiremos como cero.

Definiremos P_i y Q_i distribuciones de probabilidad condicional para los puntos de X y Y. Si Y modelara a la perfección los puntos de X, entonces $p_{j|i} = q_{j|i}$, es decir, las distribuciones elegidas fueron las más adecuadas. Como no es un caso que suceda en la vida real, debemos de suponer que P_i y Q_i son diferentes, necesitando una manera de medir qué tanto lo son, para posteriormente minimizar dicha discrepancia.

3.1.1. Divergencia de Kullback-Leibler

Teoría de la información es una rama teórica de probabilidad y estadística desarrollada principalmente por Claude Shannon a finales de la década de los 40. Poco después, en 1951, Solomon Kullback y Richard A. Leibler introdujeron el concepto de entropía relativa, o mejor conocida como la divergencia de Kullback-Leibler [8]. Se dirá que $KL(P \,||\, Q)$ mide la diferencia entre dos distribuciones de probabilidad. Usaremos dicha divergencia como función de costo:

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{3.3}$$

Muchas veces esta función se usa como si fuera una métrica, pero notemos que por una lado no satisface la desigualdad del triángulo, ni tampoco es simétrica, lo cual tendrá repercusiones que se discutirán más adelante.

Analicemos rápidamente cómo se comporta la divergencia de KL cuando es evaluada con distintos valores: el caso trivial se da cuando $P_i = Q_i$ haciendo que la función de costo sea 0. Por una lado, si tuviéramos un par de puntos x_i y x_j que estuviesen alejados entre sí, pero sus imágenes y_i y y_j fueran cercanas, el valor de C sería muy bajo, pues $p_{j|i} \leq q_{j|i}$. En el caso donde las imágenes de puntos cercanos en X se alejan en Y, C incrementaría drásticamente, pues no estaría generando una representación fiel de nuestro *input*. Es por esto que SNE trabaja bien con las estructuras locales, pues intenta preservar las distancias entre puntos, evitando que estos se alejen.

Cabe resaltar que hasta ahorita hemos pasado por alto un parámetro importante, pues este modificará nuestra percepción sobre la cercanía entre puntos. Recordando el inicio de este capítulo, se mencionó que la probabilidad de que dos puntos x_i y x_j sean vecinos se mide con respecto a una distribución Gaussiana centrada en x_i. Como vemos en la figura 3.1, el hecho de variar σ_i^2 provoca que la gráfica de una normal se ensanche o contraiga. Si lo volvemos a pensar de manera geométrica, al cambiar σ_i^2, el radio de la bola aumentará o disminuirá, creando la posibilidad para más puntos de ser vecinos de x_i.

Una de las formas en las que SNE preserva de manera eficiente estructuras locales es determinando cuántos vecinos quisiéramos que cada uno de los puntos tuviera. Así, definiremos un hiperparámetro llamado perplejidad, la cuál está definida como:

$$Perp(P_i) = 2^{H(P_i)} \tag{3.4}$$

donde $H(P_i)$ es conocida como la entropía de Shannon [9], definida por:

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \tag{3.5}$$

Es importante entender por qué estos dos conceptos nos ayudarán a obtener una medición de cuántos vecinos queremos que los puntos tengan y cómo es que están relacionados con la varianza σ_i. La entropía de Shannon mide la incertidumbre con respecto a una fuente de información [9], entre mayor sea la incertidumbre generada, mayor será la entropía. Por ejemplo, en un escenario donde la probabilidad de que ocurra algún evento es equiprobable, la entropía se maximizará, pues no hay mayor información para hacer predicciones sobre

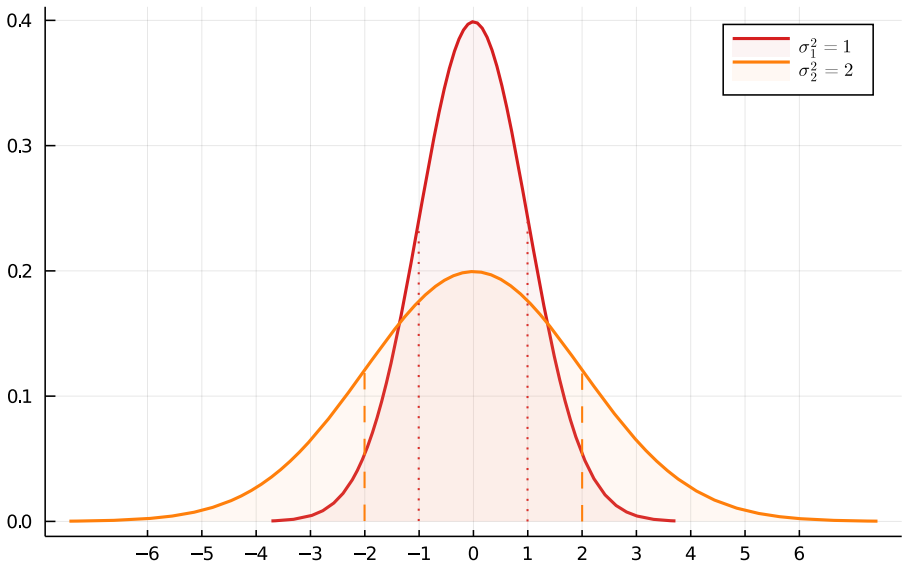


Figura 3.1: Distribución Gaussiana con distintas varianzas

sus resultados. Por otro lado, la perplejidad puede ser interpretada como una medida eficaz de los vecinos que queremos que tenga x_i [10]. Notemos que al exponencial la entropía, lo que estamos haciendo exponenciar la cantidad de información que tenemos sobre dicho evento, por lo que debería de ser el total de información que tenemos disponible, o también, lo podemos ver como un promedio con distintos pesos de las posibilidades que pueden ocurrir, como lo vemos en la ecuación 3.6.

$$2^{H(P_i)} = 2^{\sum_j p_{j|i} log_2\left(\frac{1}{p_{j|i}}\right)} = \prod_j \left[\left(\frac{1}{p_{j|i}}\right)^{p_{j|i}}\right] \tag{3.6}$$

Una vez estableciendo la perplejidad, que normalmente varia entre 5 y 50, se correrá una búsqueda binaria para encontrar la σ_i que satisfaga este valor. Más adelante se discutirán las repercusiones de variar la perplejidad y cómo esto afecta a la manera en la que se visualizan los distintos *clusters* de datos.

Previamente mencionamos que la divergencia de Kullback-Leibler medirá la diferencia entre las distribuciones P_i y Q_i, por lo que nos interesa minimizar esta función con respecto a los y's para hacer estas distribuciones lo

más similar posible. Primero encontraremos el gradiente de la ecuación 3.3, para luego poder encontrar sus mínimos locales usando un método llamado *gradient descent*, el cual explicaremos más adelante.

Los cálculos del gradiente de C se podrán ver en A, el cual quedará descrito por la siguiente ecuación:

$$\frac{\partial C}{\partial y_l} = 2 \sum_m \left(p_{m|l} - q_{m|l} + p_{l|m} - q_{l|m} \right) \left(y_l - y_m \right) \tag{3.7}$$

La interpretación que podemos dar de dicho gradiente como la fuerza que se ejerce en la dirección $y_i - y_j$ en proporción a $\left(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j} \right)$. Esto atraerá o repelerá a los puntos dependiendo de qué tan cerca o lejos estén.

3.1.2. Minimización

Desde hace varias décadas los procesos de producción en masa son cada vez más eficientes. Normalmente cuando se habla de eficiencia, nos referimos a realizar un trabajo con la menor cantidad de recursos, ya sea de alguna materia prima, energía, o el más importante, tiempo. Buscamos hacer las cosas lo más rápido posible, desde estudiar para un examen o incrementar las ganancias en algún negocio, es decir, queremos minimizar nuestro tiempo y generar mayores recompensas.

Teoría de la complejidad es una rama de estudio dentro de ciencias de la computación la cual se encarga de estudiar los recursos temporales y espaciales que una computadora necesita para procesar algoritmos. El recurso temporal es el tiempo que necesitaría la computadora para poder resolver un problema. Por otro lado, el recurso espacial es la memoria RAM que la computadora tiene disponible. Si no existiera este tipo de limitantes, no importaría qué tan rápido un algoritmo o una función resuelve un problema determinado. En los primeros semestres de la carrera se enseña a calcular los mínimos de funciones de una y de varias variables. Entre más variables tenga la función la complejidad de encontrar los mínimos globales de estas aumenta. Las funciones de costo usadas en este tipo de algoritmos generalmente manejarán grandes cantidades de variables.

Definición 3.1.1.

- Sea $f \in C^\infty(\mathbb{R})$. Se dice que f tiene un mínimo local en a si $f'(a) = 0$ y $f''(a) \geq 0$.

- Sea $f \in C^\infty(\mathbb{R}^n, \mathbb{R})$. Se dice que f tiene un mínimo en $\mathbf{a}$ si $\nabla f(\mathbf{a}) = 0$ y $\nabla^2 f(\mathbf{a})$ es semidefinida positiva.

- Sea $f(a) = b$ un mínimo de f y B el conjunto de los mínimos de f. Se dirá que b^* es un mínimo global si mín $(B) = b^*$.

Saber si un punto es mínimo de una función es bastante sencillo, solamente tenemos que corroborar que cumpla las condiciones para serlo. Por otro lado, querer saber cuáles son todos los mínimos de una función, es un problema no trivial, y que ha tenido ha muchos matemáticos con noches de desvelo. Esta es un área bastante estudiada, y si bien se han desarrollado muchos métodos que nos pueden dar mucha información al respecto, no existe uno infalible. Para ilustrar un poco, imaginemos que estamos para dos sobre una variedad en $\mathbb{R}^3$ y tenemos que encontrar al menos uno de sus mínimo. Lo primero que tendríamos que hacer es elegir una dirección en la cual caminar, igual lo que nuestra intención nos dice es que comencemos donde veamos la bajada más inclinada. Ahora supongamos que en el mejor de los casos encontramos, después de un rato, un mínimo. ¿Esto nos da algo de información? ¿Esto nos dice cuántos mínimos más tiene? O incluso, ¿sabemos si este es un mínimo local o global? También hay que contemplar que entre más grande sea la dimensión del espacio y de la variedad, más direcciones de búsqueda habrá. Si bien la cantidad de direcciones que tiene $\mathbb{R}^n$ son las mismas sin importar la n, al momento de querer pasar este problema a la práctica y pedirle a una computadora que encuentre los mínimos de una función, la complejidad del problema se acrecenta, pues la cantidad de variables a computar será mayor.

Uno de los algoritmos de optimización más usados en *Machine Learning* para minimizar las funciones de costo es *Gradient Descent*. Este algoritmo es un proceso iterativo en el cual, dado un vector, nos moveremos en dirección contraria a donde crece la función, así para encontrar un punto donde el gradiente sea cero. Hay muchísimas variaciones de esta técnica, y dependiendo del problema que estemos trabajando, una será mejor que otra. Imaginemos una superficie en $\mathbb{R}^3$ y donde elijamos el punto de inicio, ponemos una canica y la soltamos; esta parará en algún mínimo de la función. De esta forma, dado un punto x_0, *Gradient Descent* encontrará una sucesión de puntos dada por:

$$x_{n+1} = x_n - t\nabla f(x_n) \tag{3.8}$$

donde t es el tamaño de cada paso. Si bien es usual normalizar el gradiente

(pues siempre es más fácil trabajar con vectores unitarios), no es del todo necesario [11].

Elegir este escalar de manera incorrecta puede ser contraproducente para este proceso, pues si lo elegimos muy pequeño, el proceso podría tardar mucho más tiempo del necesario, aunque por otro lado, si este valor llegara a ser demasiado grande, incluso nos podríamos alejar del mínimo.

Ejemplo 3.1. *Sea* $f(x) = (x+5)^2$. *Si* $x_0 = 3$ *y* $t = .01$. *Por otro lado, si* $t = 1$, *el algoritmo se la pasaría brincando entre el valor de* $x = 3$ *y* $x = -13$.

Hay muchas variaciones de este algoritmo, pero la que SNE usa es la siguiente:

$$\mathcal{Y}_{n+1} = \mathcal{Y}_n - t\frac{\partial C}{\partial \mathcal{Y}} + \alpha(n)(\mathcal{Y}_{n-1} - \mathcal{Y}_{n-2}) \tag{3.9}$$

donde $\alpha(n)$ es un parámetro llamado *momentum*, el cual acelerará drásticamente la búsqueda del mínimo, en especial en curvas cuyo gradiente es cercano a cero. Cada iteración acumulará un poco de ese *momentum* (asemejándose a una bola cayendo por un bajado poco inclinada), haciendo que el algoritmo brinque rápidamente esos momentos de poca pendiente [11].

Intuitivamente podemos hablar de minimizar la función de costo mediante el reacomodo de los elementos de Y, es decir, después de cada iteración iremos moviendo todos los y_i de tal forma que las distancias entre ellos se asimilen lo mejor posible a las de sus contrapartes en X.

3.2. t-SNE

En la sección anterior se describieron los pasos que sigue SNE así como la teoría que lo sustenta. Si bien SNE logra generar una visualización de los datos bastante buena, este algoritmo presenta un problema que será explicado con detalles más adelante, llamado *Crowding Problem*. Hinton y van der Maaten [10] desarrollaron una versión que mejora substancialmente a SNE modificándolo de dos maneras:

1. Se actualizó la función de costo C que usamos por una versión simétrica, de esta manera optimizando los tiempos de computabilidad.

2. En vez de modelar la similitud entre dos puntos de Y con una distribución Gaussiana, se usará la distribución t de Student.

En el apéndice A se minimizó la suma de la divergencia de Kullback-Leibler entre dos distribuciones de probabilidad condicional P_i y Q_i, pero como alternativa para hacer simétrica esta función, es posible reducir C a solo un factor de la suma previa de la divergencia de KL, pero ahora será entre distribuciones de probabilidad conjuntas P y Q:

$$C = KL\left(P\|Q\right) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{3.10}$$

y se define a p_{ii} y q_{ii} como cero. Respondiendo al primer cambio para desarrollar una versión simétrica de SNE, es recordar que lo minucioso de calcular el gradiente fue que los índices no conmutaban, es por eso que se definirá a

$$q_{ij} = \frac{\exp\left(-\left\|y_i - y_j\right\|^2\right)}{\sum_{k \neq l} \exp\left(-\left\|y_k - y_l\right\|^2\right)}. \tag{3.11}$$

La forma intuitiva para definir a p_{ij} es hacerlo de forma similar

$$p_{ij} = \frac{\exp\left(-\left\|x_i - x_j\right\|^2 / 2\sigma^2\right)}{\sum_{k \neq l} \exp\left(-\left\|x_k - x_l\right\|^2 / 2\sigma^2\right)}$$

pero esto causaría que algún x_i aislado, su distancia con respecto a otros puntos fuera muy grande, y por lo tanto, sus valores p_{ij} muy chicos, generando un efecto mínimo en la función de costo, y por lo tanto, en su imagen en Y. Por esto, se definirá mejor como

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}. \tag{3.12}$$

El gradiente de esta nueva función de costo es

$$\frac{\partial C}{\partial y_i} = 4 \sum_j \left(p_{ij} - q_{ij}\right)\left(y_i - y_j\right).$$

3.2.1. *Crowding problem* y la infinidad de soluciones

Minimizar el gradiente de la función de costo provoca que los puntos de Y se acomoden de tal manera que la diferencia entre P y Q sea la menor posible; si bien esta diferencia podría ser cero, la realidad es que con datos de la vida real es prácticamente imposible. Ya vimos que modelar las distancias entre puntos de 2-variedades es bastante eficaz, así como el conjunto S y

otros ejemplos juguete, pero aumentar la dimensión de estas variedades y de los espacios donde están encajadas complica más el problema.

El *crowding problem* ocurre con los problemas de reducción de dimensión que pretenden preservar estructuras locales, así como SNE y *Sammon mapping* [10], [12]. Este problema consiste en que no se pueden preservar fielmente las distancias dentro de una vecindad de puntos, debido a que la dimensión intrínseca de estos afecta la forma en la que son medidos. Por ejemplo, en $\mathbb{R}^m$ es posible modelar un conjunto A de $m+1$ puntos de tal forma que equidisten entre sí, a este poliedro se le conoce como un m-simplejo, pero modelar este mismo conjunto en $\mathbb{R}^2$ no es posible, pues uno de los puntos no equidistaría de los demás.

En 2007, Cook intentó arreglar este problema [13] con un método llamado UNI-SNE, que si bien generaba mejores visualizaciones que SNE, tenía un gran problema: optimizar la función de costo no funcionaba debido a que si en una iteración dos puntos se separaban un poco, no habría manera de volverlos a juntar, pues esto tendría un efecto minúsculo en sus proyecciones.

Así como en la versión simétrica de SNE transformamos las distancias euclideanas de X y Y y las convertimos en probabilidades, una manera natural de atacar el *crowding problem* sería usando esta misma lógica. En X seguiremos convirtiendo las distancias a probabilidades usando la distribución Gaussiana, pero en Y lo haremos con una distribución de probabilidad que tenga la cola más pesada que una Gaussiana, esto significa, que la distribución no esté acotada en la manera en la que decrece por una exponencial, así como lo vemos en la figura 3.2. t-SNE emplea una distribución t-Student con un grado de libertad, la cual es la misma que una distribución de Cauchy. La probabilidad conjunta q_{ij} se definirá cómo:

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}} \tag{3.13}$$

Esta distribución tiene propiedades computacionales y teóricas que beneficiarán a t-SNE. Una de estas es que $\left(1 + \|y_i - y_j\|^2\right)^{-1}$ se aproxima a la ley inversa del cuadrado para distancias suficientemente grandes [14]. Teóricamente la distribución t-Student es similar a la distribución normal, en el sentido que esta es una mezcla de una infinidad de Gausianas. Por otro lado, computacionalmente es más eficiente usar las distribuciones t-Student, pues en su forma analítica no involucra exponenciales, aunque esta sea una mezcla infinita de distribuciones normales.

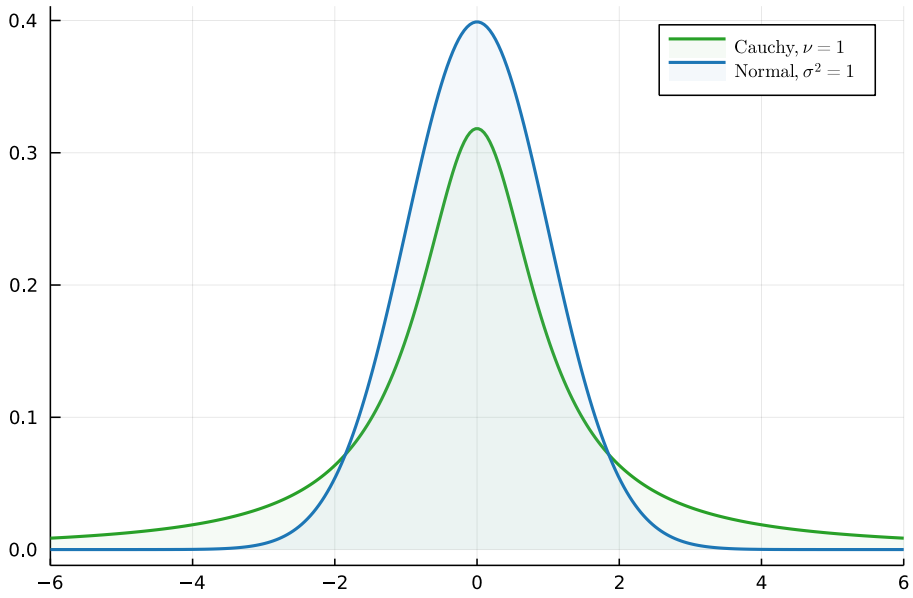


Figura 3.2: Student's t-distribution

El cálculo del gradiente de la función de costo usando esta nueva distribución de probabilidad se encuentra en el apéndice A de [10]. Su expresión analítica es:

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij}) (y_i - y_j) \left(1 + \|y_i - y_j\|^2\right)^{-1} \tag{3.14}$$

Sigue manteniendo una expresión bonita al igual que en SNE, generando un juego de fuerzas que atraen y repelen.

3.3. UMAP

El gran problema al que se enfrenta la reducción de dimensión es mantener la estructura de los datos lo mejor posible. El camino que t-SNE usa es conociendo la densidad de los puntos del conjunto X, generando distribuciones de probabilidad y midiendo su diferencia usando la divergencia de Kullback-Leibler con respecto a las distribuciones de probabilidad de un conjunto Y de puntos generados aleatoriamente en $\mathbb{R}^2$, para después minimizarla

con un proceso de *gradient descent*. Tanto t-SNE como UMAP aplican métodos usados en coloración de gráficas para generar un diseño óptimo en los datos del espacio bajo-dimensional. Por un lado, debido a la forma del gradiente de KL, existe una fuerza de repulsión entre todos los puntos de Y. Por el otro, UMAP traduce las propiedades geométricas y topológicas de la variedad subyacente M de tal forma que en vez de trabajar con un espacio continuo y complicado, lo hace con un espacio combinatorio, discreto y sencillo. Así, construirá una gráfica con los puntos de X, y posteriormente optimizará una gráfica con los puntos de Y para ser estructuralmente lo más parecidos.

3.3.1. Creando una gráfica de múltiples dimensiones

El primer paso para entender de UMAP es la manera en que el algoritmo construye la gráfica en el espacio alto-dimensional. Usaremos varias imágenes en esta sección, todas tomadas de [15], para ilustrar la teoría. Comenzaremos tomando el siguiente ejemplo mostrado en la figura 3.3. Pensemos que tenemos un conjunto de datos cuya variedad subyacente es la gráfica de la función seno. Como vemos, hay regiones que son más pobladas que otras, siendo parte de la estructura geométrica que quisiéramos preservar.

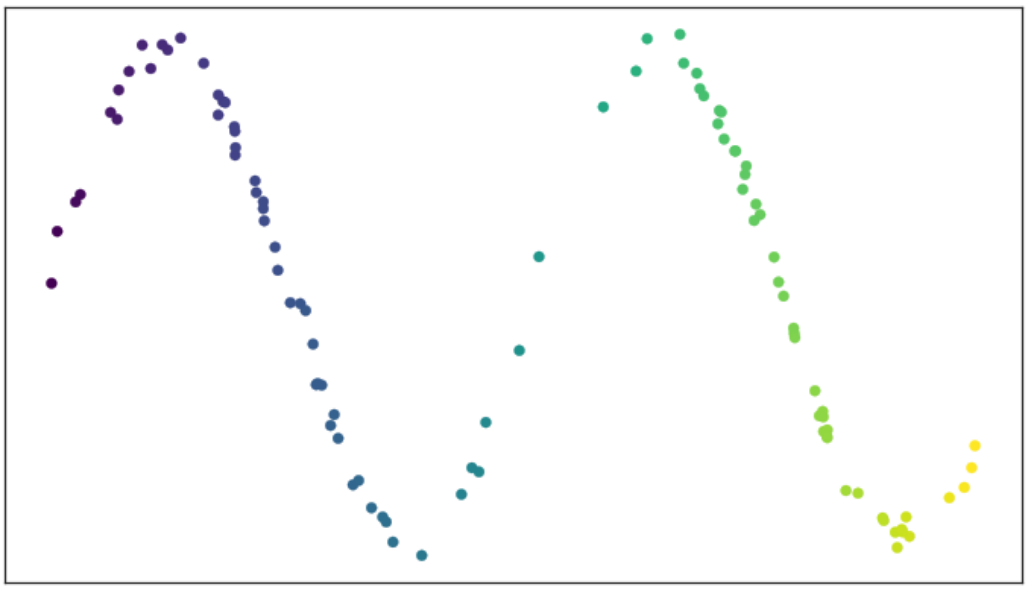
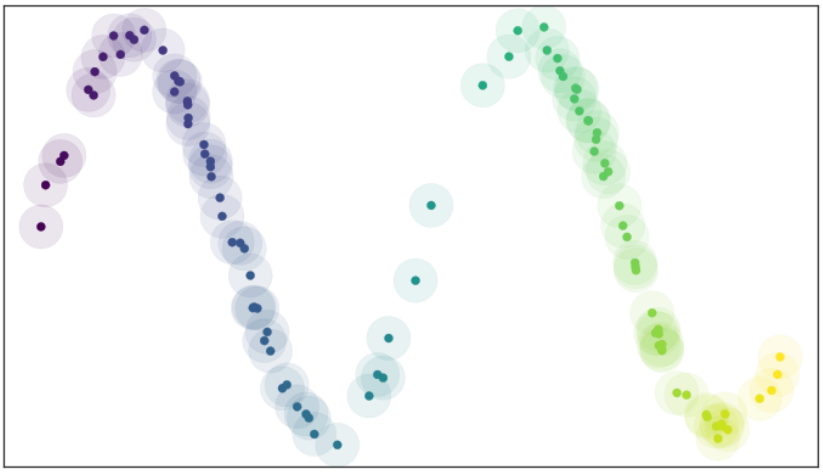
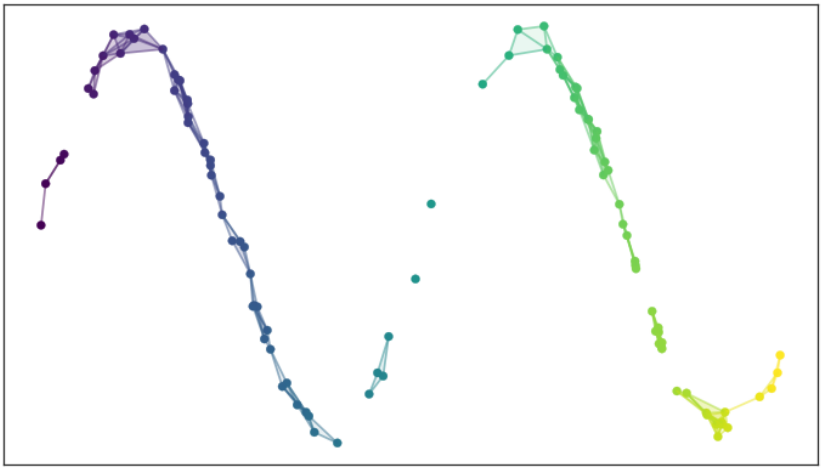


Figura 3.3: Conjunto de datos generados alrededor de la función seno.

Recordando cómo funciona t-SNE, necesitamos generar un concepto de cercanía entre los puntos, por lo que fijaremos un radio r y dibujaremos una cubierta abierta de bolas, como se ve en la figura 3.4a. Si cada vez que dos de estos abiertos se intersecten ponemos una arista entre sus centros, generaremos una gráfica de nuestro conjunto. En este caso, podemos ver que la estructura se preserva de buena manera, aunque la gráfica obtenida es inconexa y tiene algunos puntos aislados, como vemos en la figura 3.4b.



(a) Generando una cubierta abierta.



(b) Creando la gráfica de la variedad.

Figura 3.4

Aquí es donde UMAP hace una suposición un poco irreal, pero que nos ayudará a ver el problema de la reducción de dimensión desde otros ojos. Si pensamos en el escenario donde un conjunto de datos tiene sus puntos distribuidos uniformemente con respecto a su variedad adyacente M, podríamos generar una cubierta abierta de X que refleje de mejor manera su estructura geométrica y topológica, como vemos en la figura 3.5. Como esto no sucede naturalmente, forzaremos con matemáticas a que sí pase.

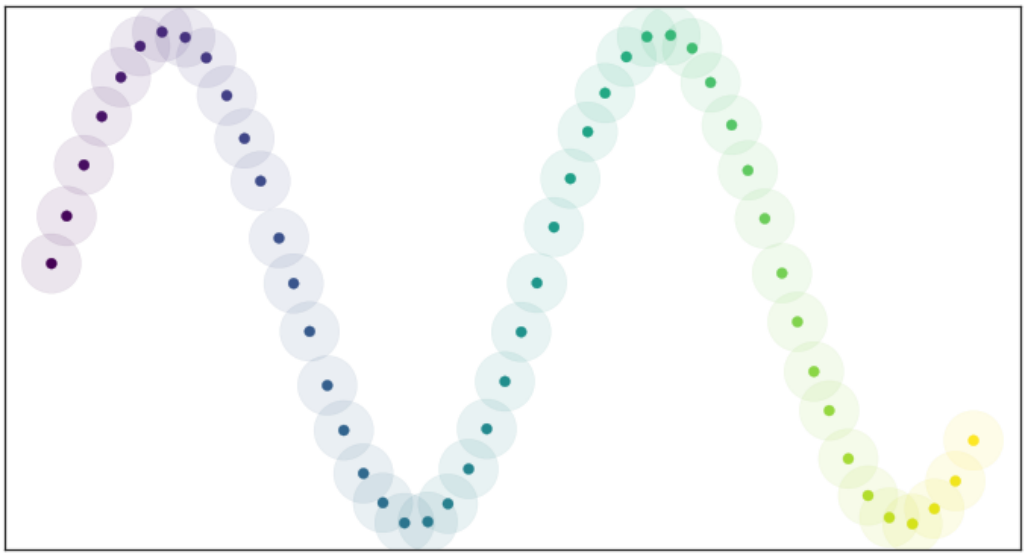


Figura 3.5: Puntos distribuidos uniformemente sobre la variedad.

La noción de que nuestros puntos estén uniformemente distribuidos se basará en que cada uno de estos tenga la misma cantidad de vecinos. Así, lo que se hace es definir una métrica de manera puntual y equipar a M con esta. Así, lo que estamos haciendo es equipar una variedad con una métrica Riemanniana g, tal que en cada punto $p \in M$ podemos definir un producto interior en el espacio tangente T_pM [16], obteniendo una noción de distancia en la variedad. Tal vez a simple vista los puntos no se vean uniformemente distribuidos, pero como cada punto tiene su propia noción de distancia[5], con respecto a la variedad M sí lo están. Podemos generar una cubierta de bolas con las distintas métricas, y como cada bola es del mismo tamaño, cada punto tendrá la misma cantidad de vecinos. Este será un hiperparámetro del algoritmo que hay que considerar, similar a la perplejidad en t-SNE. Enunciaremos el siguiente lema [17]:

Lema 3.2. *Sea (M, g) una variedad Riemanniana en un espacio ambiente $\mathbb{R}^n$, y sea $p \in M$. Si g es localmente constante alrededor de p en una vecindad abierta U tal que g es una matriz diagonal constante con coordenadas del espacio ambiente, entonces en una bola $B \subseteq U$ centrada en p con volumen $\frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)}$ con respecto a g, la distancia geodésica de p a un punto $q \in B$ es $\frac{1}{r} d_{\mathbb{R}^n}(p, q)$, donde r es el radio de la bola en el espacio ambiente y $d_{\mathbb{R}^n}$ es la métrica del espacio ambiente.*

En pocas palabras, este lema nos permitirá relacionar la métrica de nuestra variedad con respecto a la de nuestro espacio ambiente. De esta manera, la diferencia entre las figuras 3.5 y 3.6 es que la segunda, a pesar de que no vemos las bolas del mismo tamaño, sí lo están desde el punto de vista de M.

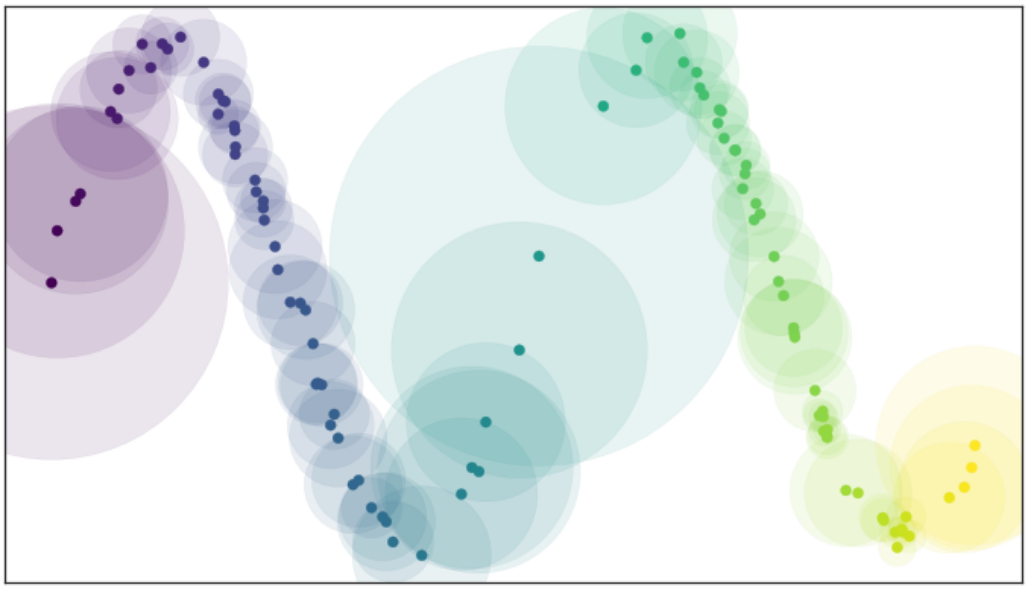
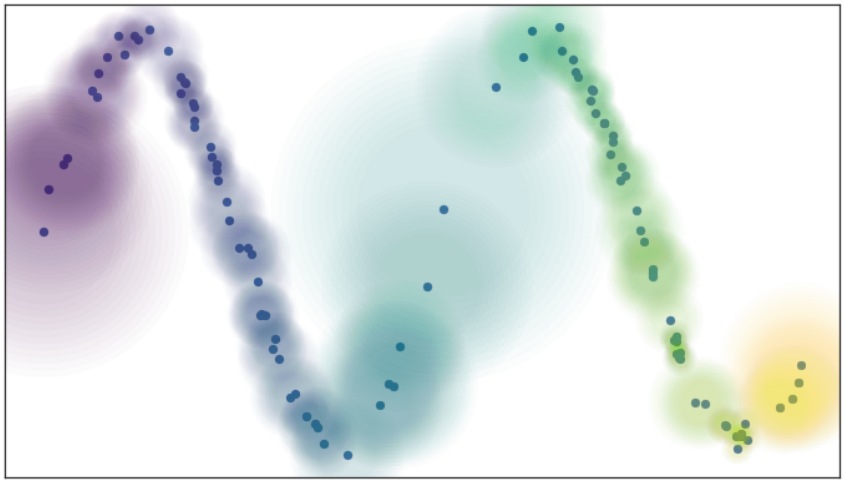


Figura 3.6: Representación de las bolas unitarias con las métricas de cada punto.

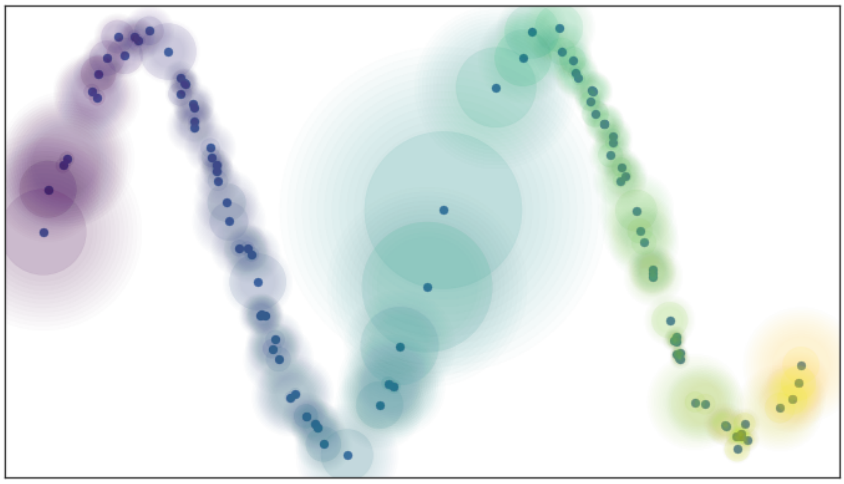
Lo que hemos logrado es poder generar una gráfica alto-dimensional, donde los k vecinos de un punto x_i estarán conectados con él. Por lo aprendido en t-SNE, esto no nos servirá del todo, ya que queremos que entre más alejados estén los puntos, menos probabilidad tengan entre sí de ser vecinos. Es decir, queremos dejar de ver la construcción de esta gráfica como un problema

[5] Faltará corroborar que todas estas métricas sean compatibles.

binario (estar o no dentro de la vecindad de un punto) y poder asignar una arista con un cierto peso (o que la probabilidad de pertenecer a la vecindad disminuya entre más lejos se encuentren). Esto se puede expresar haciendo bolas difusas, i.e., que entre más cercano se encuentre un punto a la frontera de ésta, menos probabilidad de ser vecino tendrá, así como se ve en la figura 3.7a



(a) Cubierta de bolas difusas.



(b) Cubierta de bolas difusas, con la condición

Figura 3.7

Al dejar de trabajar con este paradigma binario de pertenencia, quisiéramos que todo lo hecho anteriormente no se desperdicie, por lo que queremos evitar que esto llegara a generar puntos aislados. Así, lo que querríamos es que cada punto tenga al menos un vecino, como se ve en la figura 3.7b. Definiremos

$$\rho_i = \min\{d(x_i, x_{i_j}) | 1 \leq j \leq k, d(x_i, x_{i_j}) > 0\} \tag{3.15}$$

y una función w de pesos para las aristas

$$w((x_i, x_{i_j})) = \exp\left(\frac{-\max\left(0, d(x_i, x_{i_j}) - \rho_i\right)}{\sigma_i}\right). \tag{3.16}$$

Así, UMAP tendrá la condición de generar una gráfica que sea localmente conexa.

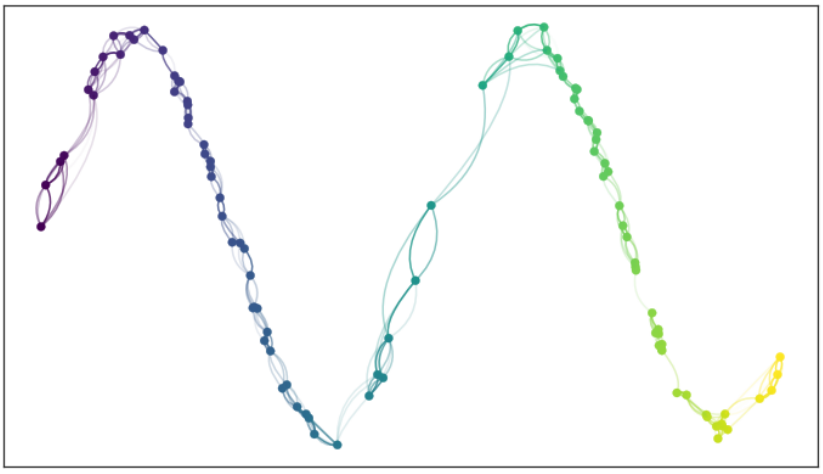


Figura 3.8: Gráfica con aristas múltiples de distintos pesos.

Todavía falta un problema por resolver: la compatibilidad de las métricas. Tenemos dos nociones de distancia distintas entre un punto p y otro q. Desde la perspectiva de p, q está a una distancia, y según q, p está a otra distinta. Lo que esto provoca son múltiples aristas entre dos puntos con pesos totalmente distintos, como vemos en la figura 3.8. Para poder unificar las aristas existen varios métodos, pero el que UMAP elige es combinar los pesos como:

$$w_1 + w_2 - w_1 w_2 = w_1(1 - w_2) + w_2. \tag{3.17}$$

Esto se puede interpretar como la probabilidad de que alguna de las dos aristas exista.

Finalmente, podemos en ver en la figura 3.9 como UMAP logra generar una representación discreta y sencilla (computacionalmente hablando) de un espacio continuo y complicado,.

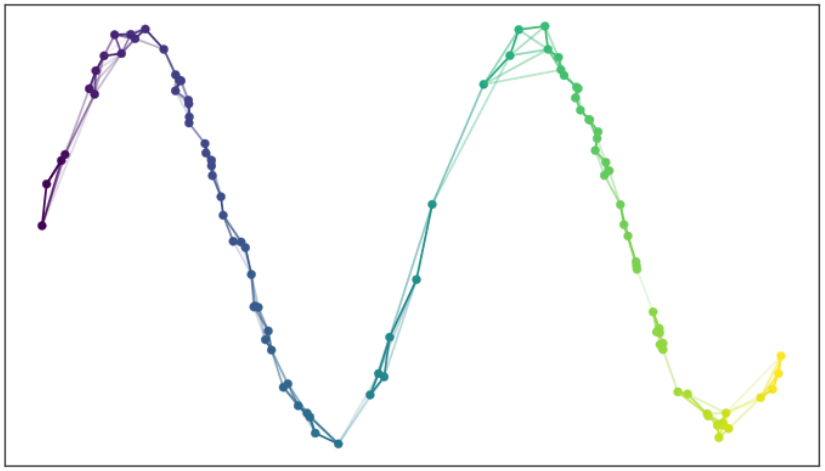


Figura 3.9: Gráfica final con métricas compatibles.

3.3.2. Optimización

Para empezar la reducción de dimensión vía UMAP, debemos de saber a qué variedad queremos llevar nuestros datos y con qué métrica está equipado dicho espacio. Como nuestro interés es la visualización de datos, sólo nos enfocaremos en $\mathbb{R}^2$ o $\mathbb{R}^3$ con sus métricas usuales como espacios de llegada. Hay que recalcar que una de las condiciones que nos gustaría mantener de la construcción que hicimos en nuestro espacio alto-dimensional es la conexidad local, es decir, garantizar que al menos existirá una arista con un peso fuerte entre cualquier vértice y un vecino de este. Por eso definiremos un hiperparámetro llamado `min_dist`.

De manera similar a t-SNE, definiremos un conjunto de puntos Y que serán nuestra contraparte bajo-dimensional. La función de costo que usaremos será la entropía cruzada y tendrá la siguiente expresión:

$$\sum_{e \in E} w_h(e) \log\left(\frac{w_h(e)}{w_l(e)}\right) + (1 - w_h(e)) \log\left(\frac{1 - w_h(e))}{1 - w_l(e)}\right) \tag{3.18}$$

E es el conjunto de todas las aristas, w_h es la función de pesos en el espacio alto-dimensional y w_l es la función de pesos en el espacio bajo-dimensional. Esta función de costo se minimizará con respecto a $w_l(e)$. Notemos que el primer término de la suma representa un fuerza de atracción entre los puntos de la arista e; este se minimizará cuando $w_l(e)$ sea lo más grande posible, por lo que los puntos se acercarán. Con la misma lógica, el segundo sumando representa una fuerza de repulsión entre los puntos de e; este se minimiza cuando $w_l(e)$ es lo más pequeño posible.

Así, podremos encontrar, al igual que en t-SNE, un juego de fuerzas entre los datos donde se atraen y repelen al mismo tiempo, logrando que la representación bajo-dimensional se optimice "moviendo"los puntos de Y.

Capítulo 4

Reducción de dimensión

Algo importante al trabajar en ciencia de datos es saber elegir bien los hiperparámetros de los modelos que serán usados. Esta comparación, además de poder ver las diferencias estructurales que t-SNE y UMAP generan en distintos conjuntos de información, servirá también para poder apreciar, en estos ejemplos particulares, si los valores elegidos fueron los adecuados, y si en algún momento llegan a generar resultados no deseados.

Usaremos dos bases de datos como objetos de estudio para probar ambas técnicas. Por un lado tenemos MNIST, la cual es una base de datos conformada por imágenes de 28 x 28 pixeles de dígitos escritos a mano, por lo que cada imagen es representada por un vector de 784 dimensiones, donde cada entrada tiene un valor entre 0 y 255 de una escala de grises. Esta fue creada en 1998 y es usada como uno de los estándares para medir la exactitud con la que distintos métodos llevan a cabo el reconocimiento de imágenes y de caracteres escritos por humanos. La segunda base de datos fue creada por Max Noichl [18] con modelos tomados del programa de escaneo 3D del *Smithsonian Institute*. La elección de dichos conjuntos de datos fueron, por un lado, usar uno de los estándares en el área para la evaluación de técnicas, y también, para evaluar la efectividad de separar datos por categorías bien definidas de forma no supervisada. De igual manera, poder hacer una evaluación de la preservación de estructuras locales y globales de los datos.

Estas simulaciones fueron corridas en Python 3.9.2, usando la versión de t-SNE integrada a la paquetería `scikit-learn` en su versión 0.24.2. En el caso de UMAP usé la versión 0.5.1, la cual no está integrada a `scikit-learn`, pero depende fuertemente de esta, la cual además tiene integrada la paquetería `Numba` para mejorar el desempeño. Los valores de los hiperparámetros fueron

fijados para mantener la consistencia en las simulaciones, definidos como perp, nn[6] = $\{5, 15, 30, 50, 100\}$. Para varias de las simulaciones hechas, se usó min_dist = .1 como valor predeterminado, de igual manera que como está escrito el algoritmo.

4.1. Reduciendo MNIST

MNIST consta de $70,000$ datos, de los cuales $60,000$ son usados como entrenamiento del algoritmo (en este caso de UMAP), para generar una versión discreta en $\mathbb{R}^2$ de la variedad subyacente a los datos (en este caso, en $\mathbb{R}^{784}$). Posteriormente se dará el resto de los datos para poder modelarlos a partir del resultado obtenido en el entrenamiento. Podemos ver en la figura 4.1 la comparación entre t-SNE y UMAP aplicado a MNIST para distintos valores de perplejidad y *nearest neighbors*, con min_dist = .1. Vemos que si bien ambos algoritmos pueden separar de manera efectiva a los dígitos, UMAP logra definir de mejor manera cada uno de los *clusters*, incluso con valores bajos de vecinos cercanos. En la figura 4.2 se hace una comparación entre distintos parámetros de UMAP, variando sobretodo min_dist, generando menor densidad de los *clusters* de cada dígito. Podemos notar algo importante en las imágenes de la primera columna con valores de $nn = 50$ y 100: al centro de la imagen, el conjunto azul deja de estar tan definido y termina esparciéndose, acercándose al *cluster* morado, situación que no pasa con los demás valores de nn. Esto es el resultado de aumentar el número de vecinos cercanos, manteniendo al mismo tiempo una distancia mínima muy chica, provocando que UMAP quiera juntar puntos que tienen ciertas similitudes, sin realmente ser el mismo dígito (por ejemplo, como el 3 y el 8). Al observar la siguiente columna vemos que esta dispersión es mucho menor, y así sucesivamente al hacer que los *clusters* tengan menos precisión en su definición.

[6]Por simplicidad, nos referiremos a la perplejidad como perp y a los *neirest neighbors* como nn

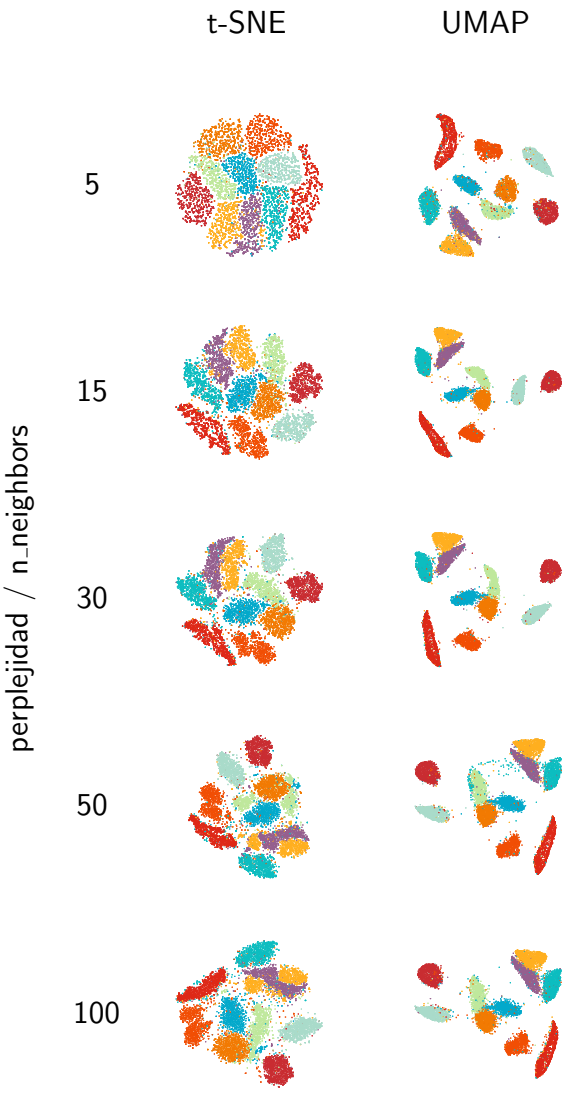


Figura 4.1: Tabla de comparación MNIST.

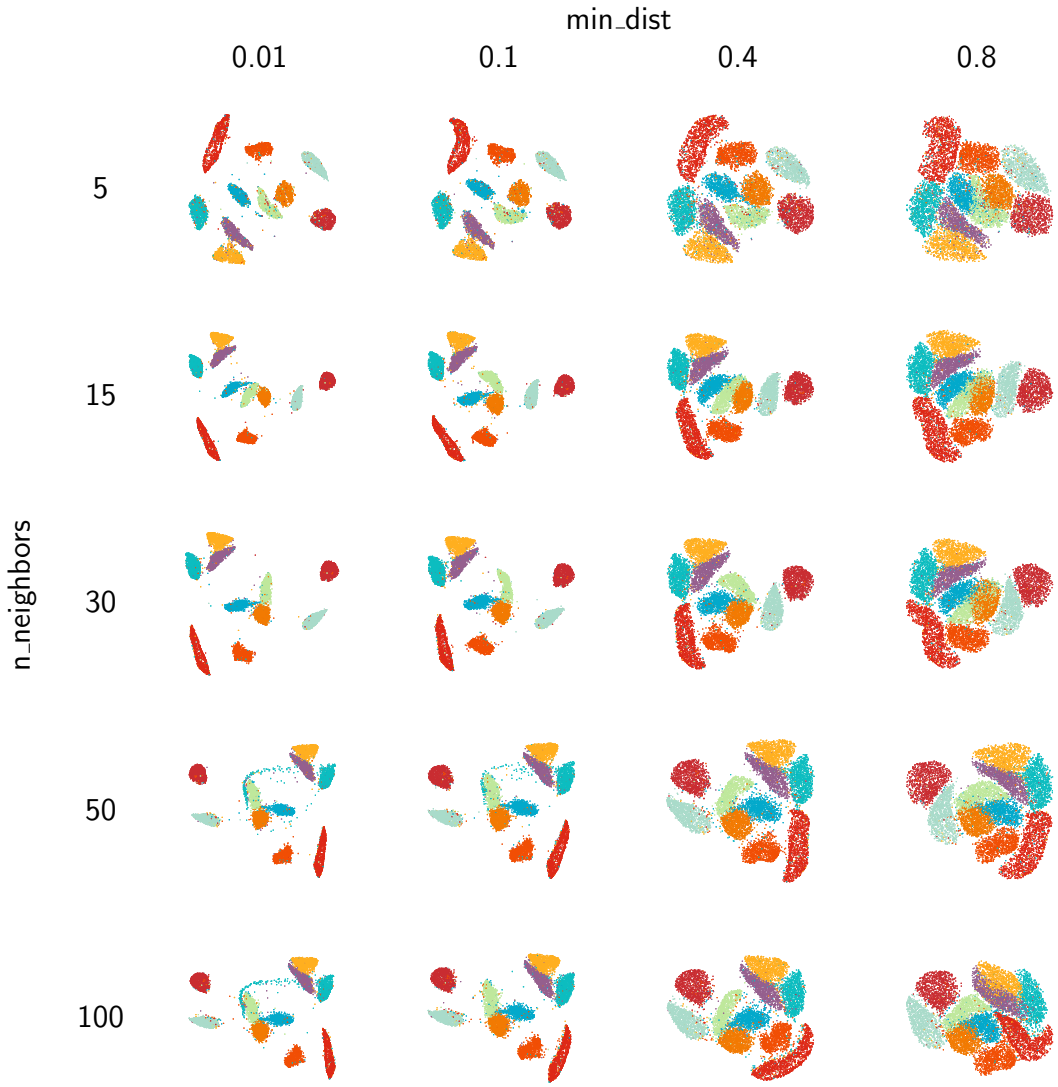


Figura 4.2: MNIST bajo múltiples hiperparámetros de UMAP.

4.2. Reduciendo Rex

La segunda base de datos, a la cual nos referiremos como Rex, consta de 499470 puntos, cada uno siendo un arreglo tridimensional. Por motivos de ejecución, las simulaciones fueron hechas con un subconjunto de $20,000$ puntos tomados aleatoriamente. Podemos ver en la figura 4.3 una toma lateral de cómo se ve el conjunto. Una de las razones por las que me interesó usar esta base de datos fue para estudiar más de cerca la preservación de las estructuras locales, es decir, si ambos métodos lograrán mantener la morfología del esqueleto y la posición de los huesos, y por otro lado, si podrán preservar las estructuras globales, diferenciando entre el Tiranosaurio rex del Triceratops siendo devorado. En este caso no se usó un conjunto para entrenar a UMAP previamente, sino que el entrenamiento se llevó a cabo con el mismo conjunto. Antes de ver las imágenes de los encajes logrados, sería valioso comparar los tiempos de ejecución de los algoritmos en el cuadro 4.1. Podemos observar que UMAP actúa más eficiente que t-SNE de manera considerable.

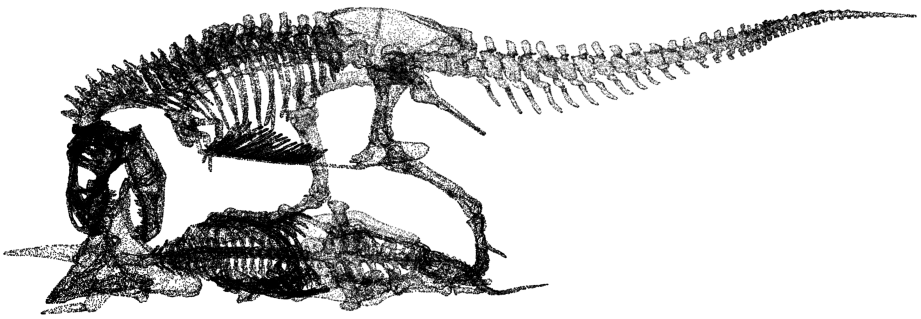


Figura 4.3: Perspectiva lateral de Rex

En la figura 4.4 podemos ver el modelado 3D de Rex, coloreándolo en 10 regiones distintas para poder apreciar la forma en la que t-SNE y UMAP

llevarán a cabo los encajes. En la figura 4.5 veremos una comparación rápida de la actuación de los algoritmos ante Rex.

segundos	perplejidad/ n_neighbors	t-SNE	UMAP
	5	46.76	13.07
	15	58.34	9.14
	30	63.18	9.65
	50	93.49	11.25
	100	102.89	18.84

Cuadro 4.1: Tiempo de ejecución t-SNE vs UMAP con Rex.

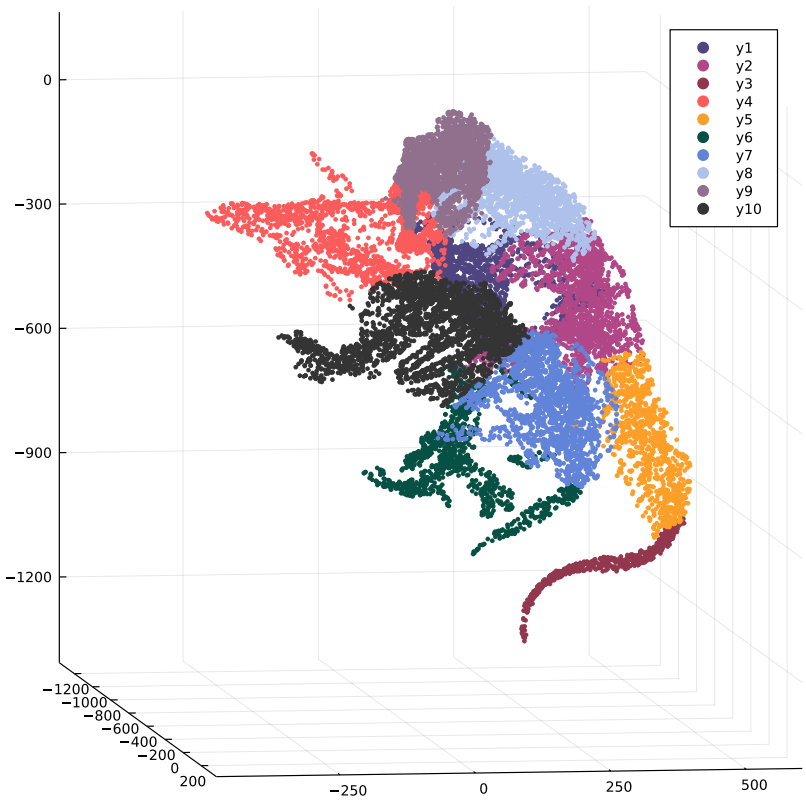


Figura 4.4: Modelo 3D de Rex.

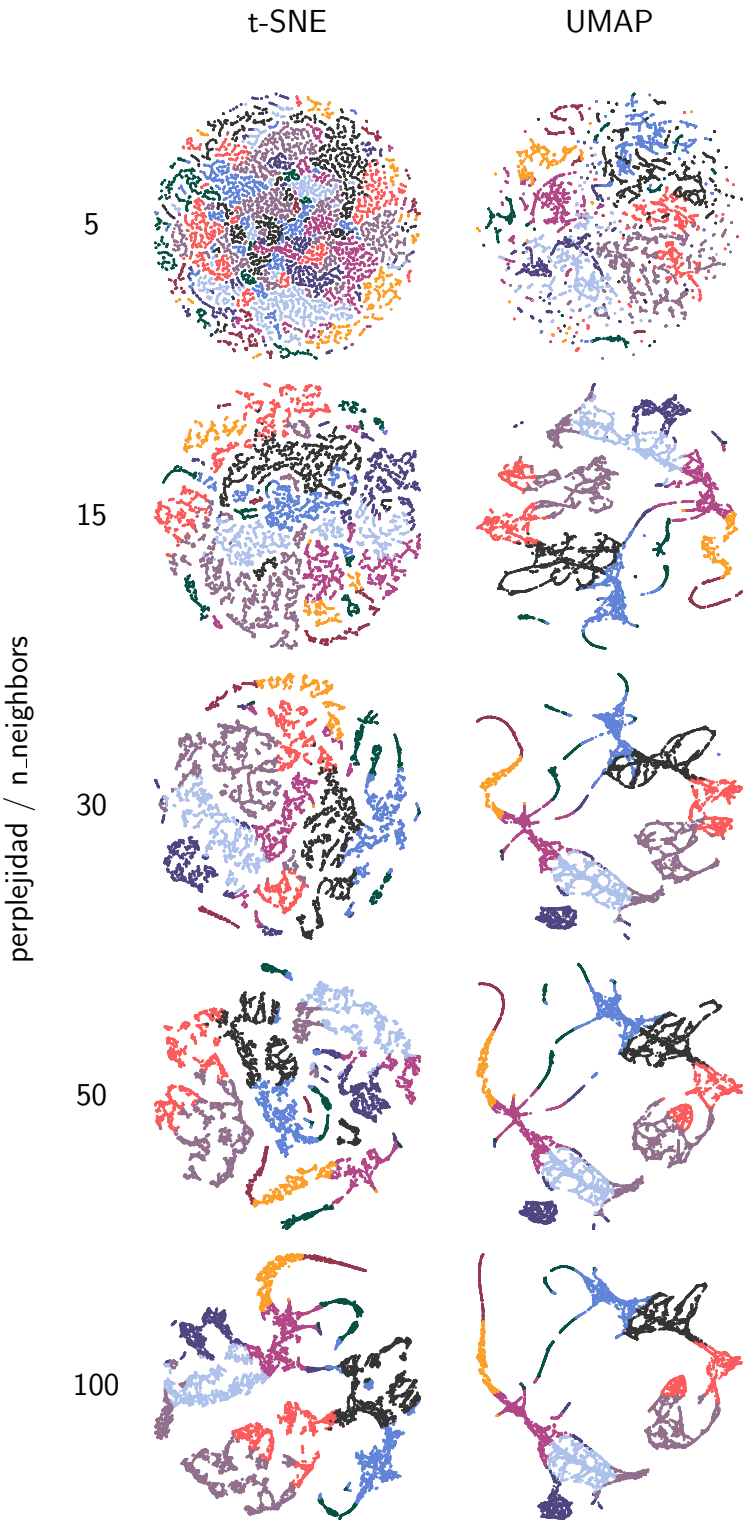


Figura 4.5: Tabla de comparación de Rex.

Cuando perp, nn = 5, tanto t-SNE como UMAP obtuvieron un encaje con muchas componentes conexas y puntos aislados. Al correr la simulación de UMAP marcó una advertencia sobre que la variedad generada podría no tener la propiedad de conexidad local debido al valor de nn. Vale la pena observar es que incluso en este primer intento, ambos logran obtener que los colores se mantengan cercanos, aunque estos se mantengan dispersos. Con perp, nn = 15 vemos que t-SNE logra definir de mejor manera los *clusters*, en cambio UMAP da un brinco radical, logrando separar ambos animales y mostrando estructuras locales. Para los siguientes valores, t-SNE sigue en el proceso de la separación de ambos animales, pero no es hasta que la perplejidad es 100 que podemos ver un cambio significativo. En cuanto a estructuras locales, que realmente es donde t-SNE destaca, observamos que logra mantener mejor las estructuras de los huesos. UMAP, además que tempranamente otorga resultados globales muy buenos, con tiempos de procesamiento muy bajos, no hace un mal trabajo obteniendo estructuras locales, sin embargo podemos notar que es mucho más agresivo al querer compactar los *clusters*, similar a como lo vimos con MNIST.

A continuación dejaré en las siguientes páginas los resultados de Rex para que el lector pueda analizar de mejor manera los encajes.

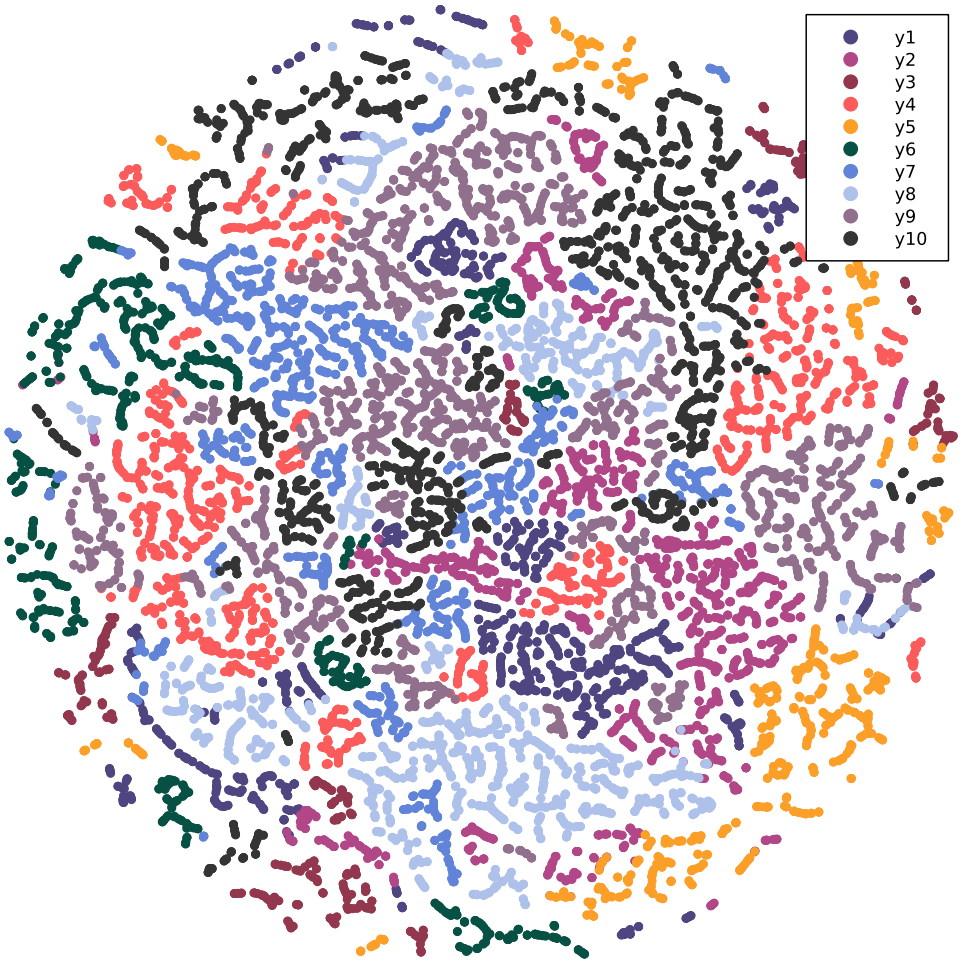


Figura 4.6: t-SNE, Perplejidad = 5

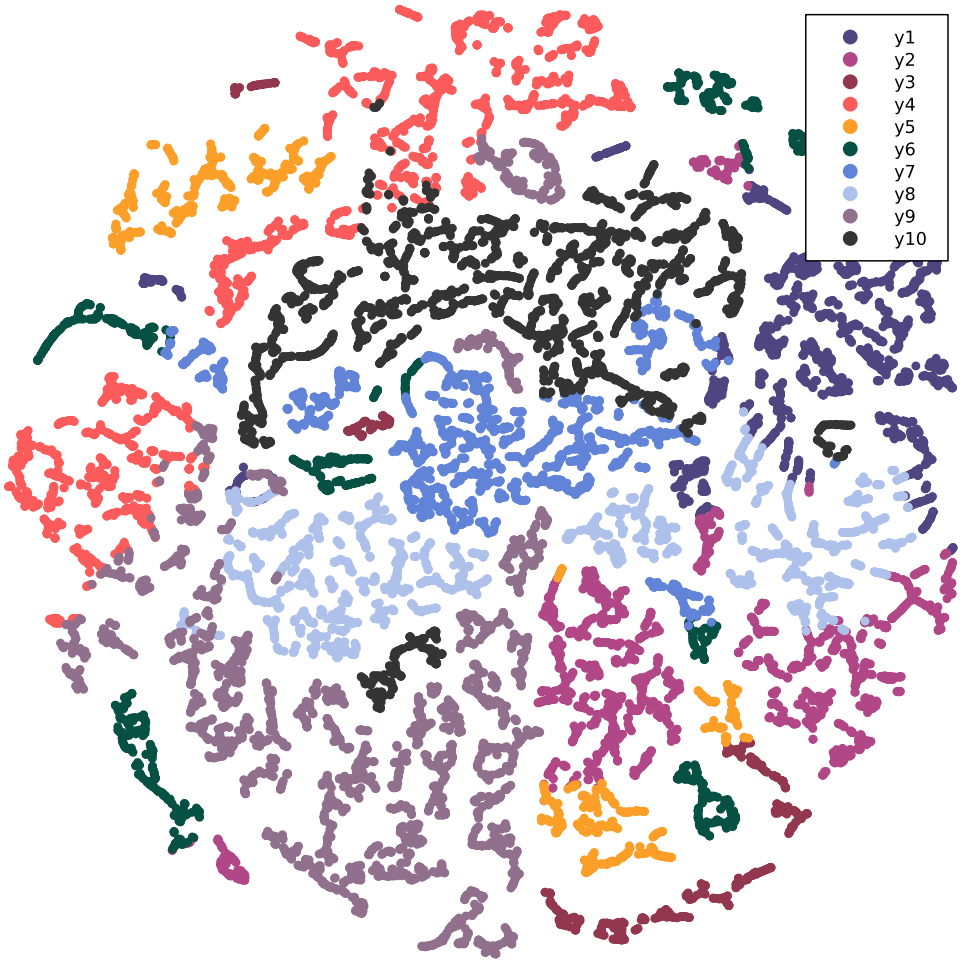


Figura 4.7: t-SNE, Perplejidad =15

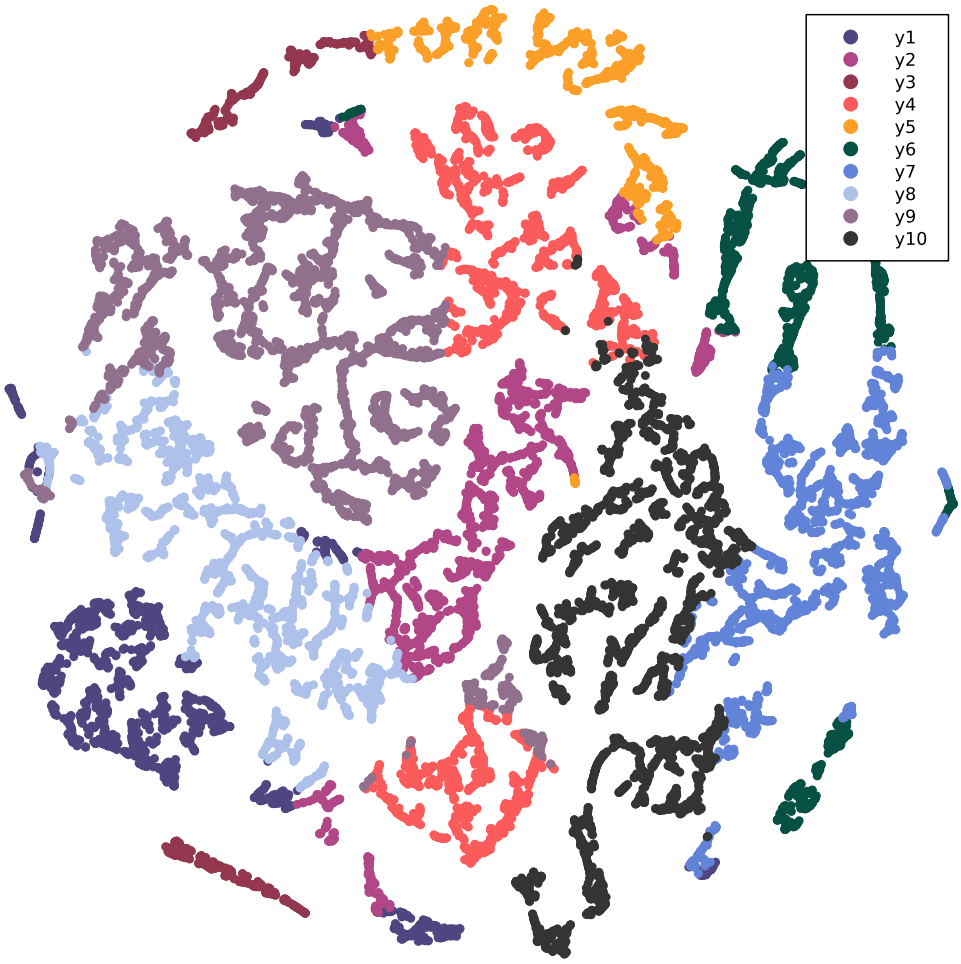


Figura 4.8: t-SNE, Perplejidad = 30

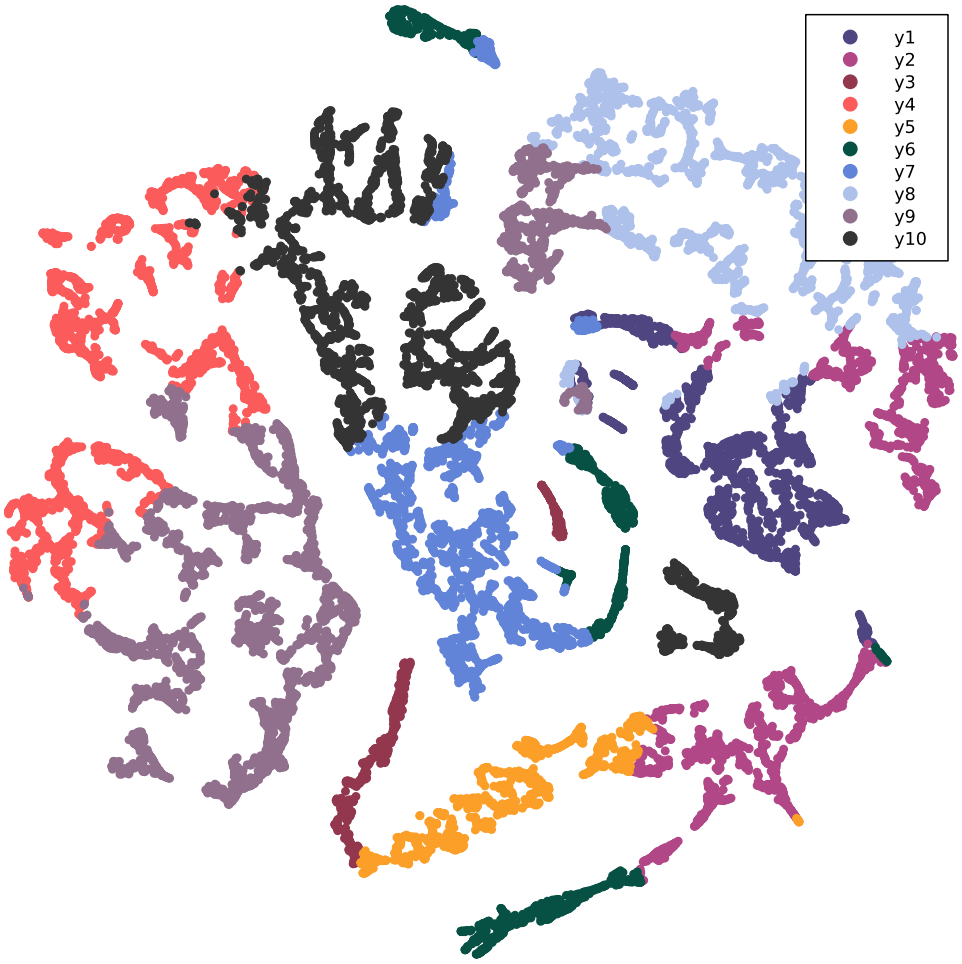


Figura 4.9: t-SNE, Perplejidad = 50

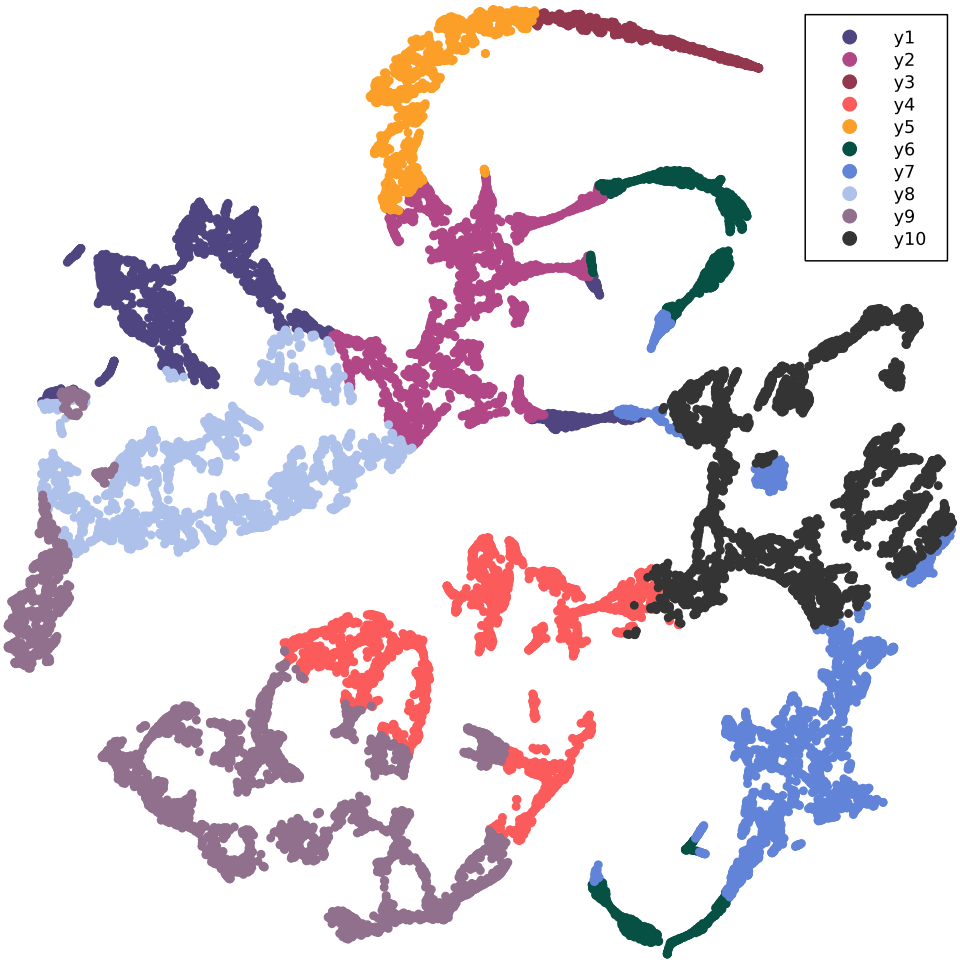


Figura 4.10: t-SNE, Perplejidad = 100

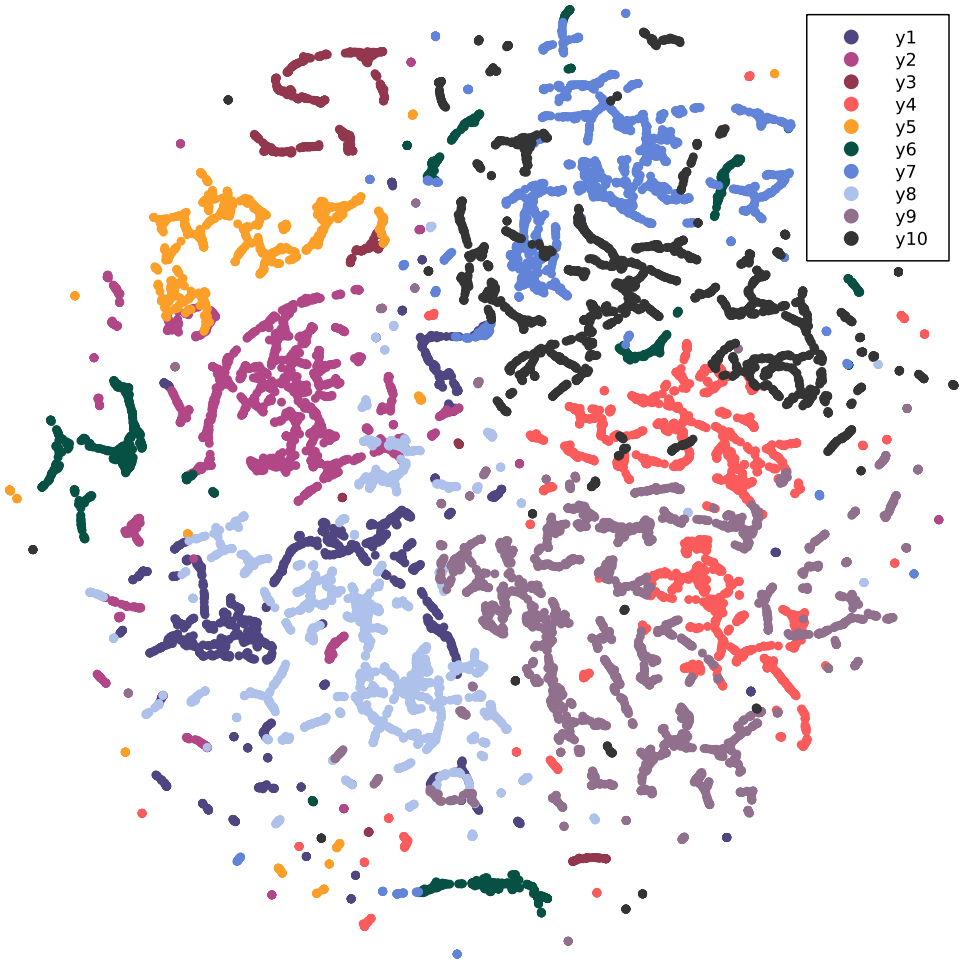


Figura 4.11: UMAP, n_neighbors = 5, min_dist = 0.1

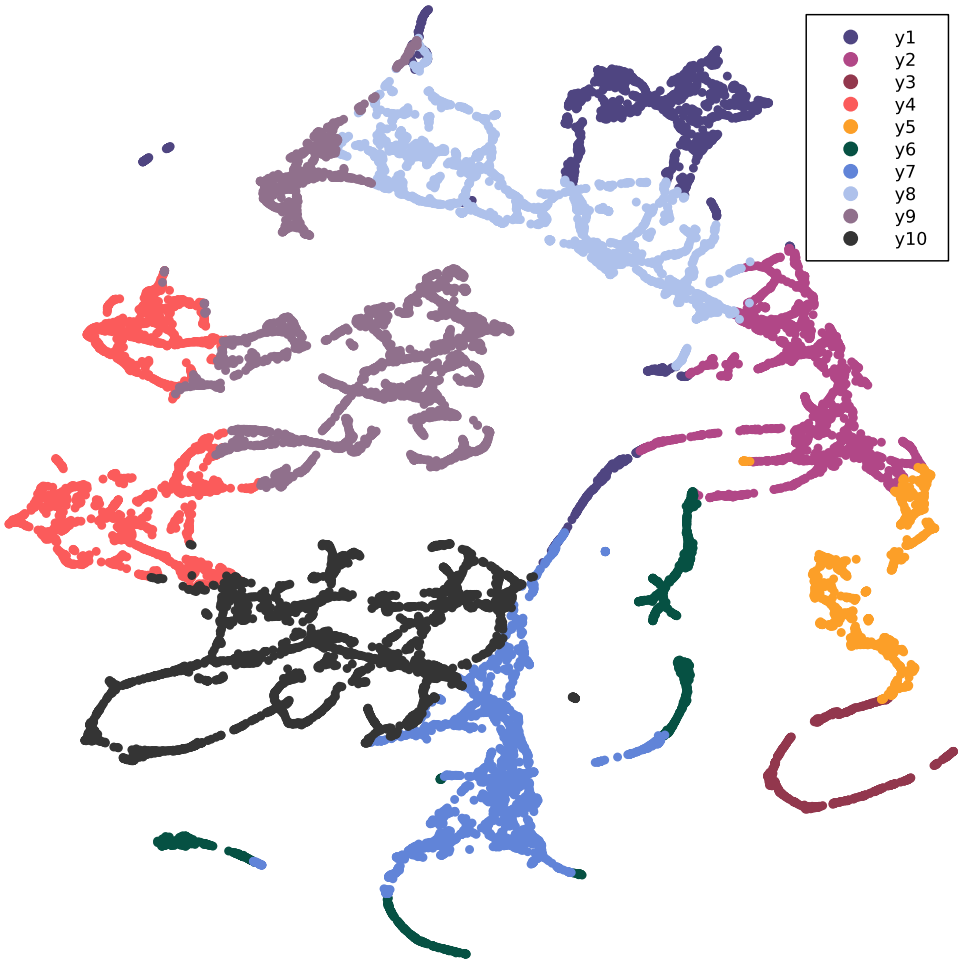


Figura 4.12: UMAP, n_neighbors = 15, min_dist = 0.1

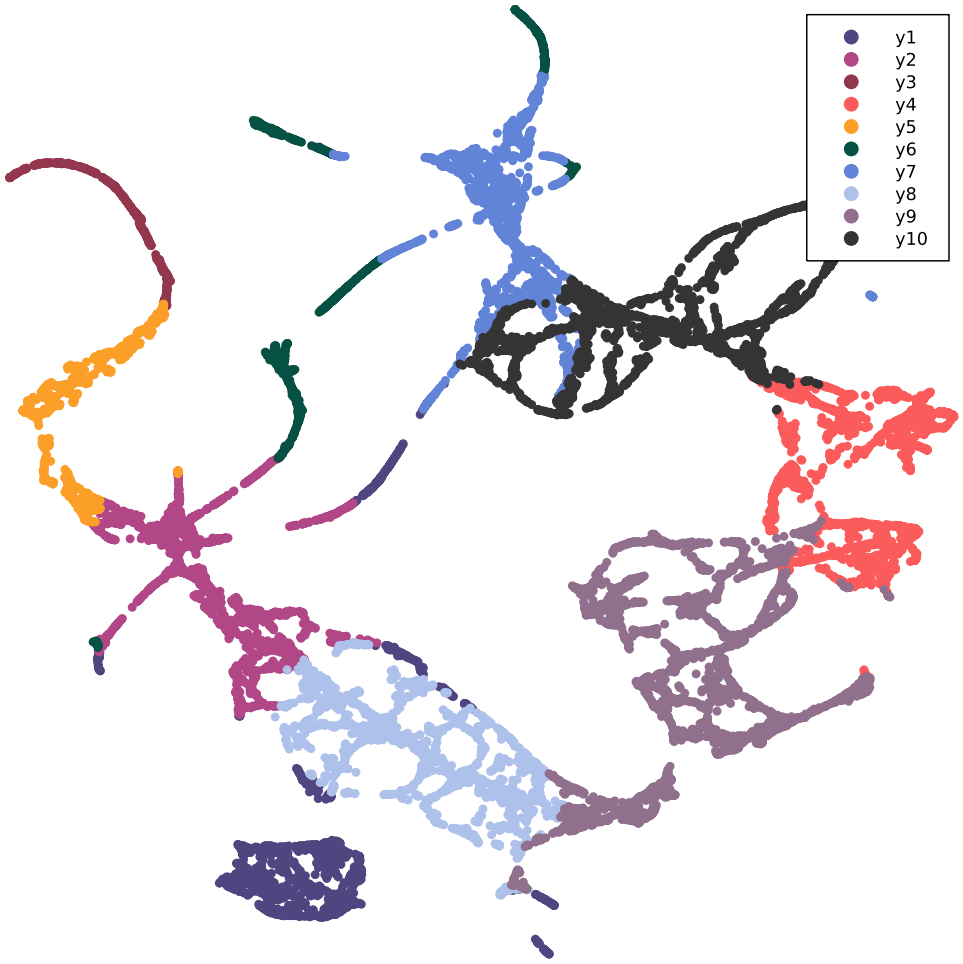


Figura 4.13: UMAP, n_neighbors = 30, min_dist = 0.1

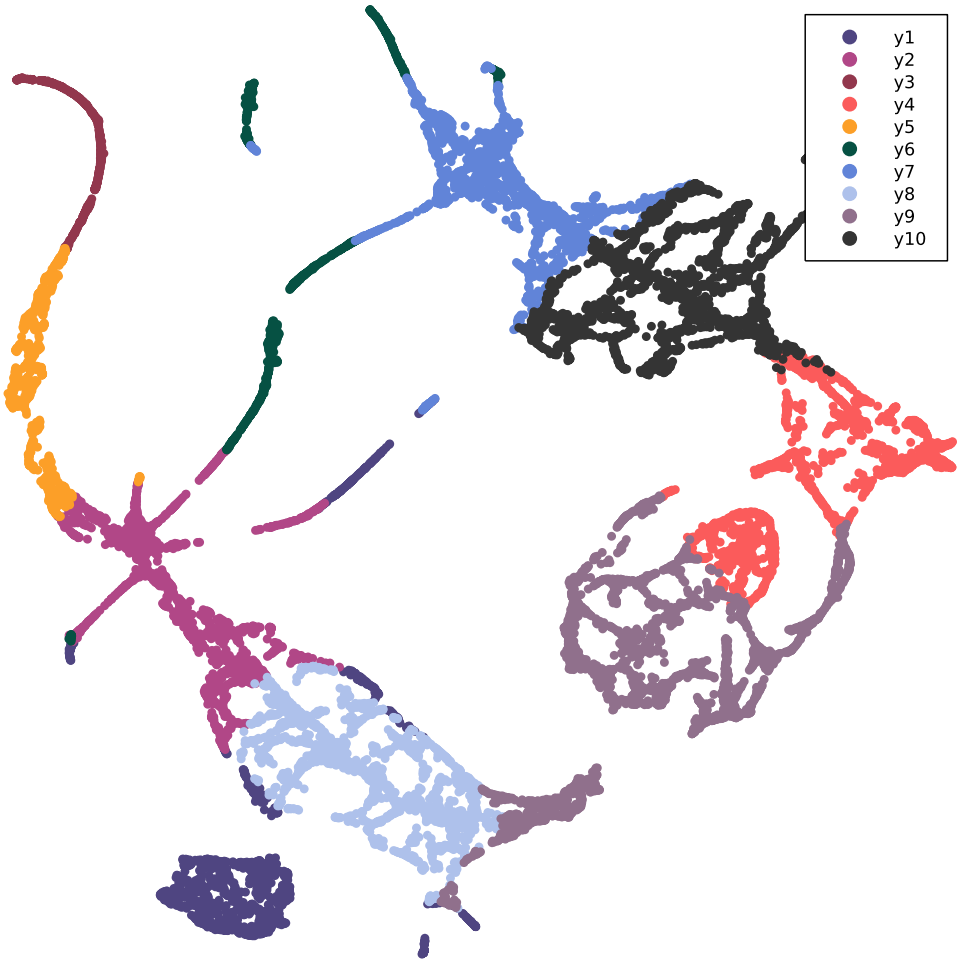


Figura 4.14: UMAP, n_neighbors = 50, min_dist = 0.1

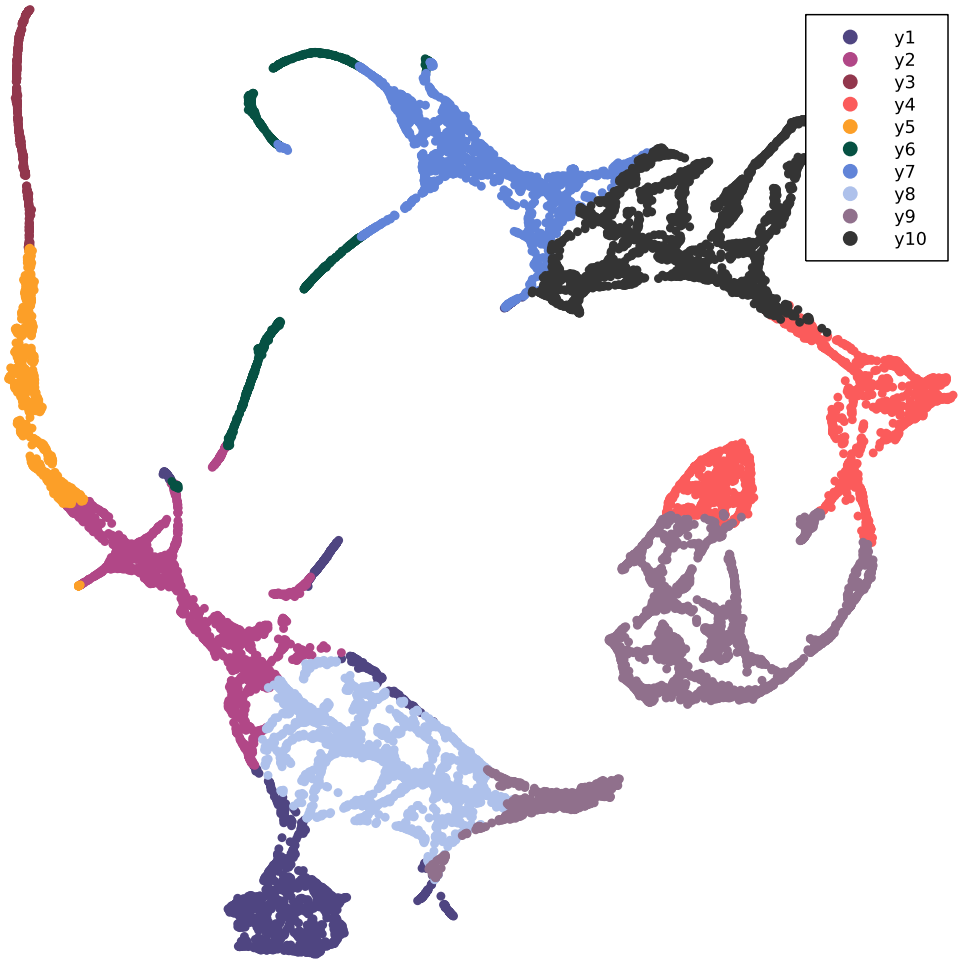


Figura 4.15: UMAP, n_neighbors = 100, min_dist = 0.1

Capítulo 5

Conclusiones

Al principio de este trabajo comenzamos hablando sobre la importancia que ha tenido la visión en la evolución del ser humano, principalmente la capacidad de encontrar patrones y poder agrupar objetos, obtenida como una consecuencia de la supervivencia. Estas habilidades las usamos en nuestro día a día, tal vez no de manera consciente, y nos ayudan constantemente para realizar tareas con cualquier complejidad. Ahora podemos confirmar lo que se comentó en los capítulos previos sobre el beneficio que obtenemos al visualizar nuestros resultados, en lugar de solamente hacer un análisis y que sea una abstracción más en nuestra cabeza.

Tanto t-SNE como UMAP han sido modelos que han revolucionado los procesos de reducción de dimensión no lineales, obteniendo una mayor precisión y eficacia. No debemos de pasar por alto que dichos algoritmos dependen enormemente del usuario y de su experiencia para que operen de la mejor manera posible, debido a que no hay una regla general que determine cuáles son los valores óptimos de los hiperparámetros. Por un lado, cada técnica tiene una sensibilidad distinta al cambio de estos valores, es decir, que los parámetros entre un modelo y otro no tienen una relación determinada, por lo que solo trabajando y experimentando con los modelos es que uno lo conocerá a fondo. Por el otro, hay que recordar que las técnicas de *Machine Learning* se basan principalmente en problemas de optimización, por lo que uno deberá de conocer cómo y en qué casos funciona mejor uno u otro algoritmo para minimizar las funciones de costo.

En los resultados del capítulo 4, podemos ver cómo operan estas técnicas y cómo compiten una con la otra. Es claro que UMAP es superior en sus tiempos de ejecución, haciéndolo 5 veces más rápido según los resul-

tados con Rex. También pudimos notar cómo con valores bajos de *nearest neighbors* logró separar de manera más clara los conjuntos de información a comparación de t-SNE. Sin embargo, estas acciones mucho más agresivas de clusterización provocan que las estructuras locales pierdan su morfología (favoreciendo en mayor medida a las estructuras globales), e incluso que datos que son suficientemente parecidos, aunque sean de clusters distintos se mezclen, así como lo vimos con MNIST. Si bien t-SNE se tarda más en cuanto a tiempo de procesamiento y sus valores de perplejidad son más altos para alcanzar resultados satisfactorios, logra preservar las estructuras locales de mejor forma, y al mismo tiempo, obtenemos un producto considerablemente bueno a nivel global. Dicho esto, enfatizamos que cada problema al que nos enfrentemos es único y dependerá tanto de la experiencia del usuario como del tipo de estudio y resultados que se quieran obtener.

Apéndice A

SNE

Recordemos que la función de costo que usará SNE es una suma de la divergencia de Kullback-Leibler, descrita en la ecuación 3.3. Comenzaremos usando la regla de la cadena para múltiples variables:

$$\frac{\partial C}{\partial y_l} = \sum_{m \neq l} \frac{\partial C}{\partial d_{ml}} \cdot \frac{\partial d_{ml}}{\partial y_l} = \sum_{m} \frac{\partial C}{\partial d_{ml}} \cdot \frac{\partial d_{ml}}{\partial y_l}. \tag{A.1}$$

Empezaremos por calcular $\frac{\partial C}{\partial d_{ml}}$, haciendo el siguiente cambio de variable que simplificará las cuentas: $d_{ji} = \|y_i - y_j\|^2$.

$$\begin{aligned}\frac{\partial C}{\partial d_{ml}} = \frac{\partial}{\partial d_{ml}} \left(\sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \right) &= \sum_i \sum_j \frac{\partial}{\partial d_{ml}} \left(p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \right) \\ &= - \sum_i \sum_j \frac{p_{j|i}}{q_{j|i}} \cdot \frac{\partial}{\partial d_{ml}} \left(q_{j|i} \right).\end{aligned}$$

Notemos que d_{ml} es simétrica debido a cómo corren los índices, por lo que consideraremos los siguientes casos:

$$\frac{\partial}{\partial d_{ml}} \left(q_{j|i} \right) = \begin{cases} \neq 0, \text{si } \begin{cases} i = l, j = m \\ i = l, j \neq m \\ i = m, j = l \\ i = m, j \neq l \end{cases} \\ 0, \text{si } i \neq m, i \neq l \end{cases}$$

En el denominador de $q_{j|i}$ hay dos sumandos, el l-ésimo y el m-ésimo, que hacen que la derivada en los primeros casos sea distinto de cero. Es por lo mismo que en el último caso la derivada se anula. Los casos 1 y 2 son análogos a los casos 3 y 4.

Caso 1 $i = l$ y $j = m$

$$\begin{aligned}
&\frac{\partial}{\partial d_{ml}}\left(q_{m|l}\right) = \frac{\partial}{\partial d_{ml}}\left(\frac{\exp\left(-d_{ml}\right)}{\sum_{k\neq i}\exp\left(-d_{kl}\right)}\right) \\
&= \frac{\left(-\exp\left(-d_{ml}\right)\right)\cdot\left(\sum_{k\neq i}\exp\left(-d_{kl}\right)\right) - \left(\exp\left(-d_{ml}\right)\right)\cdot\left(\frac{\partial}{\partial d_{ml}}\sum_{k\neq i}\exp\left(-d_{kl}\right)\right)}{\left(\sum_{k\neq i}\exp\left(-d_{kl}\right)\right)^2} \\
&= \frac{\left(-\exp\left(-d_{ml}\right)\right)\cdot\left(\sum_{k\neq i}\exp\left(-d_{kl}\right)\right) + \left(\exp\left(-d_{ml}\right)\right)\cdot\left(\exp\left(-d_{ml}\right)\right)}{\left(\sum_{k\neq i}\exp\left(-d_{kl}\right)\right)^2} \\
&= -q_{m|l} + q_{m|l}^2.
\end{aligned}$$

Caso 2 $i = l$ y $j \neq m$

$$\begin{aligned}
&\frac{\partial}{\partial d_{m|l}}\left(q_{j|l}\right) = \frac{\partial}{\partial d_{m|l}}\left(\frac{\exp\left(-d_{j|l}\right)}{\sum_{k\neq i}\exp\left(-d_{k|l}\right)}\right) = -\frac{-\exp\left(-d_{m|l}\right)\cdot\exp\left(-d_{j|l}\right)}{\left(\sum_{k\neq i}\exp\left(-d_{k|l}\right)\right)^2} \\
&= \frac{\exp\left(-d_{m|l}\right)}{\sum_{k\neq i}\exp\left(-d_{k|l}\right)}\cdot\frac{\exp\left(-d_{j|l}\right)}{\sum_{k\neq i}\exp\left(-d_{k|l}\right)} = q_{m|l}\cdot q_{j|l}.
\end{aligned}$$

Juntando ambos casos obtenemos:

$$\begin{aligned}
&\frac{\partial C}{\partial d_{jl}} = -\sum_j \frac{p_{j|l}}{q_{j|l}}\cdot\frac{\partial}{\partial d_{jl}}\left(q_{j|l}\right) = -\frac{p_{m|l}}{q_{m|l}}\left(-q_{m|l} + q_{m|l}^2\right) - \sum_{j\neq m}\frac{p_{j|l}}{q_{j|l}}\left(q_{m|l}\cdot q_{j|l}\right)) \\
&= p_{m|l} - p_{m|l}\cdot q_{m|l} - \sum_{j\neq m} p_{j|l}\cdot q_{m|l} \;=\; p_{m|l} - \left(p_{m|l}\cdot q_{m|l}\right) - \left(1 - p_{m|l}\right)^{7}\cdot q_{m|l} \\
&= p_{m|l} - \left(p_{m|l}\cdot q_{m|l}\right) - q_{m|l} + \left(p_{m|l}\cdot q_{m|l}\right) = p_{m|l} - q_{m|l}.
\end{aligned}$$

Caso 3 y 4 $i = m$ y $j = l$ o $j \neq l$

[7] $\sum_j p_{j|i} = 1$, entonces $\sum_{j\neq m} p_{j|l} = 1 - p_{m|l}$.

Por simetría:

$$\frac{\partial C}{\partial d_{ml}} = p_{l|m} - q_{l|m}.$$

Juntando los cuatro casos tenemos:

$$\frac{\partial C}{\partial d_{ml}} = \sum_m \left(p_{m|l} - q_{m|l} + p_{l|m} - q_{l|m} \right).$$

Finalmente calculamos $\frac{\partial d_{ml}}{\partial y_l}$:

$$\frac{\partial d_{ml}}{\partial y_l} = \frac{\partial}{\partial y_l} \left(\| y_l - y_m \|^2 \right) = 2 \left(y_l - y_m \right).$$

De esta manera, obtenemos el gradiente de la función de costo de SNE:

$$\frac{\partial C}{\partial y_l} = \sum_m \frac{\partial C}{\partial d_{ml}} \cdot \frac{\partial d_{ml}}{\partial y_l} = 2 \sum_m \left(p_{m|l} - q_{m|l} + p_{l|m} - q_{l|m} \right) \left(y_l - y_m \right).$$

Bibliografía

[1] F. Simion and E. D. Giorgio, "Face perception and processing in early infancy: inborn predispositions and developmental changes," *Frontiers in Psychology*, vol. 6, p. 969, 2015.

[2] F. Farzin, C. Hou, and A. M. Norcia, "Piecing it together: Infants' neural responses to face and object structure," *Journal of Vision*, vol. 12, pp. 6–6, 12 2012.

[3] R. Zafarani, M. A. Abbasi, and H. Liu, *Social Media Mining: An Introduction*. Cambridge University Press, 2014.

[4] C. Cadwalladr and E. Graham-Harrison. `https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election`. 12 de noviembre de 2021.

[5] T. Arnold, M. Kane, and B. W. Lewis, *A Computational Approach to Statistical Learning (1st ed.)*. CRC Press, 2019.

[6] A. J. Izenman, *Modern Multivariate Statistical Techniques*. Springer, 2008.

[7] G. Hinton and S. Roweis, "Stochastic Neighbor Embedding," vol. 15, 06 2003.

[8] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[9] C. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, no. July, October, pp. 379–423, 623–656, 1948.

[10] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[11] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*. The MIT Press, 2019.

[12] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. C-18, pp. 401–409, 05 1969.

[13] J. Cook, I. Sutskever, A. Mnih, and G. Hinton, "Visualizing similarity data with a mixture of maps," *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, vol. 2, pp. 67–74, 2007.

[14] O. Gal and R. Chen-Morris. `https://articles.adsabs.harvard.edu//full/2005HisSc..43..391G/0000398.000.html`. 1 de diciembre de 2021.

[15] L. McInnes, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." `https://umap-learn.readthedocs.io/en/latest/how_umap_works.html#`. 9 de mayo de 2021.

[16] M. do Carmo, *Riemannian Geometry*. Mathematics (Boston, Mass.), Birkhäuser, 1993.

[17] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," 2020.

[18] M. Noichl. `https://github.com/MNoichl/UMAP-examples-mammoth-`. 9 de mayo de 2021.

[19] S. Kullback, *Information Theory and Statistics*. Dover Publication, 1968.