Universidad Tecnológica Nacional

Tesis de Maestría

Generación de datos sintéticos para selección de características con algoritmos genéticos

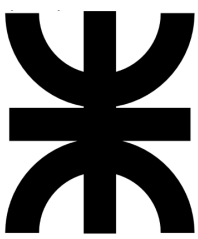
Autor:
Claudio Sebastian Castillo

Directores:
Matías Gerard y Leandro Vignolo

Tesis presentada en cumplimiento de los requisitos para el grado de Maestría en Minería de Datos

en

UTN
Seccional Paraná



Agosto, 2024

"*We can only see a short distance ahead, but we can see plenty there that needs to be done.*"

A.M.Turing, Computing Machinery and Intelligence

UNIVERSIDAD TECNOLÓGICA NACIONAL

Seccional Paraná

Maestría en Minería de Datos

Resumen

Generación de datos sintéticos para selección de características con algoritmos genéticos

by Claudio Sebastian Castillo

La disponibilidad de datos muestrales afecta a los procesos de selección de características, y resulta particularmente condicionante en escenarios de alta dimensionalidad y bajo número de muestras. En el caso de selección de características mediante AGs la falta de datos muestrales impacta negativamente en la función de aptitud, y de esa forma limita la eficacia del algoritmo. Por eso, la técnica de aumentación de datos mediante AVs plantea una posible solución a este problema, ofreciendo distintas alternativas de implementación en el contexto de los AGs.

Acknowledgements

A mis hijos y mi mujer por su amor infinito.

Table of contents

List of Figures

List of Tables

List of Abbreviations

LAH **L**ist **A**bbreviations **H**ere
WSF **W**hat (it) **S**tands **F**or

List of Symbols

a	distance	m
P	power	W (J s^1)
ω	angular frequency	rad

Para Morella, Sofía, Joquín y Manuel. Para Verónica

Capítulo 1

Introducción

Aquí digo algo importante.

Capítulo 2

Algoritmos clásicos

En este capítulo revisaremos el desempeño de algoritmos o modelos clásicos en la solución de los problemas de clasificación planteados en los dataset elegidos para nuestra investigación. A tal fin describiremos brevemente la composición de los conjuntos de datos, los algoritmos seleccionados para su tratamiento y los resultados obtenidos para cada uno. Luego, analizando y comparando dichos resultados, elegiremos los modelos con mejor desempeño en las tareas de clasificación considerando su eficacia, rapidez y consistencia a lo largo de las distintas tareas.

El propósito de esta etapa del trabajo es doble. Por un lado, identificar los modelos más apropiados para servir de *función de fitness* en la implementación de nuestro algoritmo genético. Esto permitirá construir una implementación robusta, que cuenta con una función efectiva y computacionalmente conveniente para evaluar cada solución. Por el otro, disponer de métricas acerca del desempeño que logran distintas estrategias de clasificación, a partir de las cuales comparar el resultado de nuestras propias soluciones.

2.1. Datos elegidos en nuestro estudio

El conjunto de datos elegidos en este trabajo incluye:

1. *Madelon*: conjunto artificial de datos con 500 características, donde el objetivo es un XOR multidimensional con 5 características relevantes y 15 características resultantes de combinaciones lineales de aquellas (i.e. 20 características redundantes). Las otras 480 características fueron generadas aleatoreamente (no tienen poder predictivo). Madelon es un problema de clasificación de dos clases con variables de entrada binarias dispersas. Las dos clases están equilibradas, y los datos se dividen en conjuntos de entrenamiento y prueba. Fue creado para el desafío de Selección de Características NIPS_2003, y está disponible en el Repositorio UCI. Los datos están divididos en un conjunto de entrenamiento y un conjunto de testeo.

2. *Gisette:* es un dataset creado para trabajar el problema de reconocimiento de dígitos escritos a mano (Isabelle Guyon 2004). Este conjunto de datos forma parte de los cinco conjuntos utilizados en el desafío de selección de características de NIPS 2003. Tiene 13500 observaciones y 5000 atributos. El desafío radica en diferenciar los dígitos ‘4’ y ‘9’, que suelen ser fácilmente confundibles entre sí. Los dígitos han sido normalizados en tamaño y centrados en una imagen fija de 28x28 píxeles. Además, se crearon características de orden superior como productos de estos píxeles para sumergir el problema en un espacio

de características de mayor dimensión. También se añadieron características distractoras denominadas "sondas", que no tienen poder predictivo. El orden de las características y patrones fue aleatorizado. Los datos están divididos en un conjunto de entrenamiento y un conjunto de testeo.

3. *Leukemia*: El análisis de datos de expresión génica obtenidos de micro-datos de ADN se estudia en Golub (1999) para la clasificación de tipos de cáncer. Construyeron un conjunto de datos con 7129 mediciones de expresión génica en las clases ALL (leucemia linfocítica aguda) y AML (leucemia mielogénica aguda). El problema es distinguir entre estas dos variantes de leucemia (ALL y AML). Los datos se dividen originalmente en dos subconjuntos: un conjunto de entrenamiento y un conjunto de testeo.

4. *GCM*: El conjunto de datos GCM fue compilado en Ramaswamy (2001) y contiene los perfiles de expresión de 198 muestras de tumores que representan 14 clases comunes de cáncer humano. Aquí el enfoque estuvo en 190 muestras de tumores después de excluir 8 muestras de metástasis. Finalmente, cada matriz se estandarizó a una media de 0 y una varianza de 1. El conjunto de datos consta de un total de 190 instancias, con 16063 atributos (biomarcadores) cada una, y distribuidos en 14 clases desequilibradas. Los datos están divididos en un conjunto de entrenamiento y un conjunto de testeo.

2.2. Modelos Elegidos

Para disponer de métricas de base para la comparación de nuestra solución y, al mismo tiempo, evaluar el grado de complejidad que presentan los datos incluidos en nuestro estudio hemos seleccionado una serie de modelos ampliamente usados el campo del aprendizaje automático. A fin de estandarizar la implementación de estos algoritmos hemos empleado la librería `scikit-learn` que provee abstracciones convenientes para nuestro entorno de experimentación. Los modelos elegidos son:

Modelos lineales

Los modelos lineales son un conjunto de algoritmos que predicen la salida en función de una combinación lineal de características de entrada. Son particularmente útiles cuando se espera que haya una relación lineal entre variables.

- `LDA`: Análisis Discriminante Lineal, empleado para dimensiones reducidas y asumiendo distribuciones gaussianas.
- `QDA`: Análisis Discriminante Cuadrático, similar a LDA pero con covarianzas distintas por clase.
- `Ridge`: Regresión de Cresta, empleado para tratar con multicolinealidad mediante regularización L2.
- `SGD`: Descenso de Gradiente Estocástico, estrategia central del aprendizaje automático, empleado para optimizar modelos lineales.

Modelos basados en árboles

Los modelos basados en árboles implican la segmentación del espacio de características en regiones simples dentro de las cuales las predicciones son más o menos uniformes. Son potentes y flexibles, capaces de capturar relaciones no lineales y complejas en los datos.

- `AdaBoost`: Estrategia que entrena modelos débiles secuencialmente, enfocándose en las instancias u observaciones previamente difíciles de clasificar.
- `Bagging`: Estrategia que combina predicciones de múltiples modelos para reducir la varianza.
- `Extra Trees Ensemble`: Estrategia que construye múltiples árboles con splits aleatorios de características y umbrales.
- `Gradient Boosting`: Estrategia que mejora modelos de forma secuencial minimizando el error residual.
- `Random Forest`: Estrategia basada en conjunto de árboles de decisión, cada uno entrenado con subconjuntos aleatorios de datos.
- `DTC`: Árbol de Decisión Clásico, modelo intuitivo que divide el espacio de características.
- `ETC`: Árboles Extremadamente Aleatorizados, variante de Random Forest con más aleatoriedad.

Modelos de Naive Bayes

Los modelos de Naive Bayes son clasificadores probabilísticos basados en el teorema de Bayes que presupone independencia entre las características. Son modelos de rápida ejecución y eficientes.

- `BNB`: Naive Bayes Bernoulli, se emplea para características de variables binarias.
- `GNB`: Naive Bayes Gaussiano, se emplea para distribución normal de los datos.

Modelos de vecinos más cercanos

KNN es un método de clasificación no paramétrico que clasifica una muestra basándose en cómo están clasificadas las muestras más cercanas en el espacio de características. Es simple y efectivo, particularmente para datos donde las relaciones entre características son complejas o desconocidas.

- `KNN`: K-Vecinos más Cercanos, clasifica según la mayoría de votos de los vecinos.

Modelos de redes neuronales

El Perceptrón Multicapa es un tipo de red neuronal que consiste en múltiples capas de neuronas con funciones de activación no lineales. Puede modelar relaciones complejas y no lineales entre entradas y salidas, y es altamente adaptable a la estructura de los datos.

- `MLP`: Perceptrón Multicapa, red neuronal con una o más capas ocultas.

Modelos de Máquinas de Vectores de Soporte

Las Máquinas de Soporte Vectorial son un conjunto de algoritmos supervisados que buscan la mejor frontera de decisión que puede separar diferentes clases en el espacio de características. Ofrecen alta precisión y son muy efectivos en espacios de alta dimensión y en casos donde el número de dimensiones supera al número de muestras.

- `LSVC`: Máquina de Vectores de Soporte Lineal, se emplea en espacios de alta dimensión.
- `NuSVC`: SVC con parámetro Nu, que controla el número de vectores de soporte.
- `SVC`: Máquina de Vectores de Soporte, se emplea para espacios de dimensiones intermedias y altas.

Finalmente, es preciso destacar que para el dataset GCM, que contiene 14 clases en la variable objetivo, hemos excluido modelos no compatibles o ineficientes para problemas de clasificación multiclases.

2.3. Configuración de los Modelos

Para evaluar los modelos clásicos hemos decidido su configuración a partir de la búsqueda de la mejor combinación de parámetros. A tal fin, hemos seleccionado aquellos parámetros más importantes en cada modelo y respecto de cada uno establecimos una búsqueda en grilla de sus respectivos valores. Hemos seleccionado para parámetros numéricos un mínimo de 3 valores y máximo de 20, y para no numéricos hemos decidido la configuración estándar según cada modelo. El espacio de búsqueda resultante para cada modelo puede verse en el siguiente link.

2.4. Resultados Obtenidos

En la siguiente tabla resumimos los resultos obtenidos del entrenamiento de los modelos en los dataset estudiados.

Models	Leukemia Train	Leukemia Test	Madelon Train	Madelon Test	Gisette Train	Gisette Test	GCM Train	GCM Test
LDA	0.93	0.85	0.82	0.6	1.0	0.96	-	-
QDA	1.0	0.5	1.0	0.66	1.0	0.7	-	-
Ridge	1.0	0.99	0.82	0.6	1.0	0.97	-	-
SGD	1.0	0.98	0.63	0.64	1.0	0.99	1.0	0.71
AdaBoost	1.0	0.91	0.89	0.84	1.0	0.99	-	-
Bagging	1.0	1.0	0.97	0.91	-	-	-	-
DTC	1.0	0.72	0.77	0.64	0.95	0.92	0.95	0.53
ETC	1.0	0.54	0.62	0.57	0.95	0.94	0.98	0.48
Ext.Trees.Ens.	1.0	1.0	1.0	0.71	0.99	0.99	1.0	0.57
Gradient Boost.	1.0	0.99	1.0	0.82	1.0	1.0	1.0	0.58
Random Forest	1.0	1.0	1.0	0.78	0.99	0.99	1.0	0.62
BNB	1.0	0.89	0.73	0.63	0.95	0.94	-	-
GNB	1.0	0.91	0.81	0.65	0.91	0.85	-	-
KNN	0.86	0.86	0.74	0.65	0.99	0.99	-	-
LSVC	1.0	0.99	0.78	0.62	1.0	0.99	1.0	0.62
NuSVC	1.0	1.0	1.0	0.61	1.0	0.99	0.99	0.58
SVC	1.0	1.0	1.0	0.61	1.0	0.99	1.0	0.58
MLP	1.0	0.96	1.0	0.58	1.0	0.99	1.0	0.68

Estos valores dan forma a la siguiente representación:

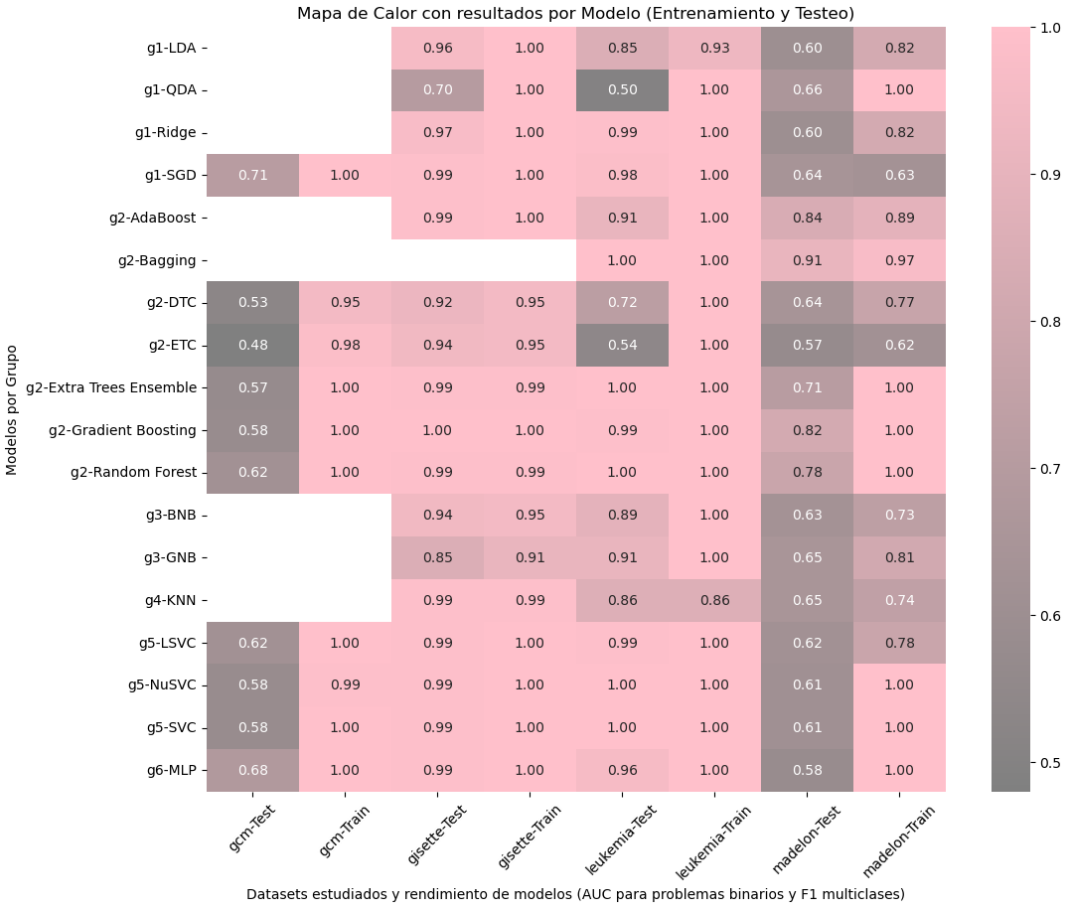


FIGURE 2.1: algoritmosclasicos

Capítulo 3

Autocodificadores Variacionales

En este capítulos presentamos la arquitectura del Autocodificador Variacional (AV) que empleamos para la generación de datos sintéticos. Exponemos brevemente sus fundamentos teóricos, los pasos que hemos seguidos en su implementación en este trabajo y las variaciones introducidas para su apropiada aplicación a los problemas abordados. En el capítulo siguiente nos enfocaremos en los Algoritmos Genéticos, sus fundamentos y características. Finalmente, el último capítulo expondremos los resultados obtenidos combinando ambas tecnologías para resolver problemas de selección de características.

3.1. Modelos generativos

Los modelos generativos (MG) son un amplio conjunto de algoritmos de aprendizaje automático que buscan modelar la distribución de probabilidad de datos observados $p_\theta(x)$. A diferencia de los modelos discriminantes (MD), cuyo objetivo es aprender un predictor a partir de los datos, en los modelos generativos el objetivo es *resolver un problema más general vinculado con el aprendizaje de la distribución de probabilidad conjunta de todas las variables* (Kingma and Welling 2019). Así, siguiendo a Kingma, podemos decir que *un modelo generativo simula la forma en que los datos son generados en el mundo real*. Dada estas propiedades, estos modelos permiten crear nuevos datos que se asemejan a los originales, y se aplican en tareas de generación de datos sintéticos, imputación de datos faltantes, reducción de dimensionalidad y selección de características, entre otros.

Los modelos generativos pueden tener como *inputs* diferentes tipos de dato, como imágenes, texto, audio, entre otros. Por ejemplo, las imágenes son un tipo de dato para los cuales los MG han demostrado gran efectividad. En este caso, cada dato de entrada x es una imagen que puede estar representada por un vector miles de elementos que corresponden a los valores de píxeles. El objetivo de un modelo generativo es aprender las dependencias (Doersch 2021) entre los píxeles (e.g. pixeles vecinos tienden a tener valores similares) y poder generar nuevas imágenes que se asemejen a las imágenes originales.

Podemos formalizar esta idea asumiendo que tenemos ejemplos de datos x, distribuidos según una distribución de probabilidad conjunta no conocida que queremos modelo $p_\theta(x)$ para que sea capaz de generar datos similares a los originales.

3.2. Autocodificadores

Los autocodificadores son un tipo de MG especializado en la representación de un espacio de características dado en un espacio de menor dimensión (de la Torre 2023). El objetivo de esta transformación es obtener una representación de baja dimensionalidad y la mayor fidelidad posible del espacio original. Para ello el modelo debe aprender una representación significativa de los datos observados, reduciendo las señales de entrada a sus dimensiones más importantes.

Los autocodificadores se componen de dos partes: un *codificador* y un *decodificador*. El *codificador* es una función que toma una observación x_i y la transforma en un vector de menor dimensión z, mientras que el *decodificador* toma el vector z y lo transforma en una observación x_i' que -idealmente- se asemeja a la observación original. Este vector de menor dimensión z es conocido como *espacio latente*.

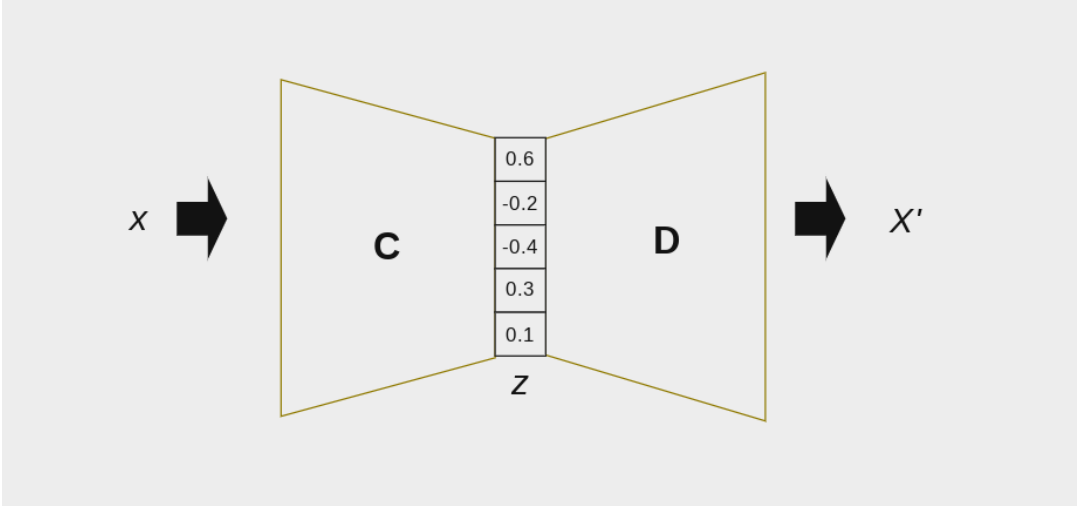


FIGURE 3.1: autocodificadores

En el proceso de aprendizaje de un autocodificador, la red modela la distribución de probabilidad de los datos de entrada x y aprende a mapearlos a un espacio latente z. Para ello, se busca minimizar la diferencia entre la observación original x_i y la reconstrucción x_i', diferencia que se denomina *error de reconstrucción*. Esta optimización se realiza a través de una *función de pérdida* que se define como la diferencia entre x_i y x_i'.

Formalmente, podemos establecer estas definiciones (de la Torre 2023):

- Sea x el espacio de características de los datos de entrada y z el espacio latente, ambos espacios son euclidianos, $x = \mathbb{R}^m$ y $z = \mathbb{R}^n$, donde $m > n$.
- Sea las siguientes funciones paramétricas $C_\theta : x \to z$ y $D_\phi : z \to x'$ que representan el codificador y decodificador respectivamente.
- Entonces para cada observación $x_i \in x$, el autocodificador busca minimizar la función de pérdida $L(x_i, D_\phi(E_\theta(x_i)))$. Ambas funciones E_θ y D_ϕ son redes neuronales profundas que se entrenan simultáneamente.

Para optimizar un autocodificador se requiere una función que permita medir la diferencia entre la observación original y la reconstrucción. Esta diferencia usualmente se basa en la *distancia euclidia* entre x_i y x_i', es decir, $||x_i - x_i'||^2$. La función de pérdida se define como la suma de todas las distancias a lo largo del conjunto de datos de entrenamiento. Tenemos entonces que:

$$L(\theta, \phi) = argmin_{\theta,\phi} \sum_{i=1}^{N} ||x_i - D_\phi(C_\theta(x_i))||^2$$

Donde $L(\theta, \phi)$ representa la función de pérdida que queremos minimizar: θ son los parámetros del codificador C y ϕ son los parámetros del decodificados D.

3.3. Autocodificadores y el problema de la generación de datos

En el proceso de aprendizaje antes descripto, la optimización no está sujeta a otra restricción mas que minimizar la diferencia entre la observación original y la reconstrucción, dando lugar a espacios latentes generalmente discontinuos. Esto sucede porque la red puede aprender a representar los datos de entrada de manera eficiente sin necesidad de aprender una representación continua. En la arquitectura del autocodificador no hay determinantes para que dos puntos cercanos en el espacio de características se mapeen a puntos cercanos en el espacio latente.

Esta discontinuidad en el espacio latente hace posible que ciertas regiones de este espacio no tengan ninguna relación válida con el espacio de características. Esto es un problema en la generación de datos, ya que la red podrá generar representaciones alejadas de los datos originales. Regularmente lo que se busca en los MG, no es simplemente una generación de datos completemante igual o totalmente distintos a los orginales, sino cierta situación intermedia donde los nuevos datos introducen variaciones en características específicas.

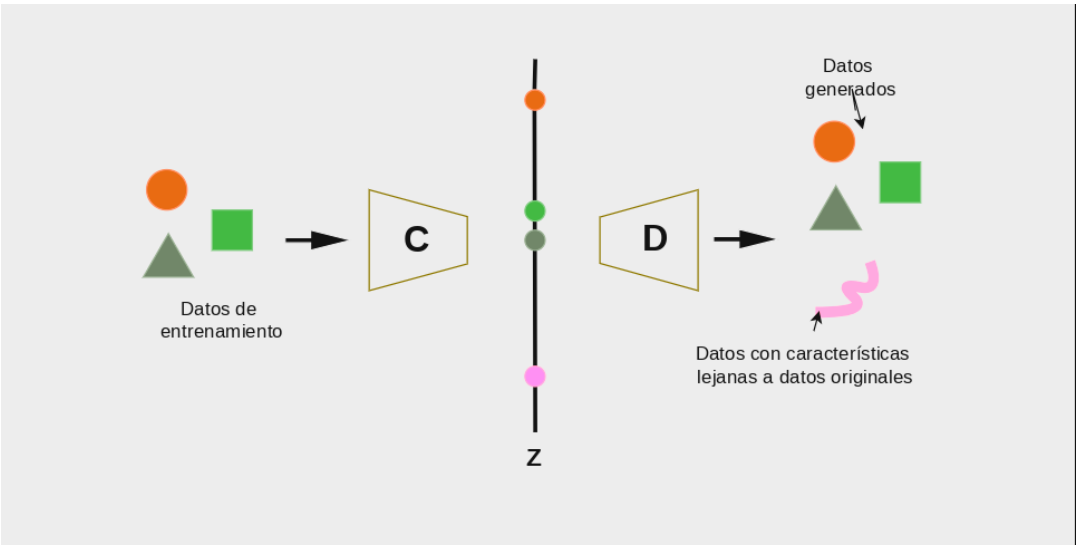


FIGURE 3.2: Discontinuidad del espacio latente

3.4. Autocodificadores Variacionales

Los Autocodificadores Variacionales (AVs) buscan resolver los problemas de discontinuidad y falta de regularidad en el espacio latente de los Autocodificadores. Comparten con éstos la arquitectura *codificador-decodificador*, pero introducen importantes modificaciones en su funcionamiento.

En esta arquitectura en lugar de realizar transformaciones determinísticas de los datos de entrada, se busca modelar la distribución de probabilidad de dichos datos aproximando la distribución *a posteriori* de las variables latentes $p_\theta(z|x)$.

En los AVs la red codificadora, también llamada *red de reconocimiento*, transforma la distribución de probabilidad de los datos de entrada en una distribución de probabilidad -generalmente más simple (e.g. distribucion normal multivaridada)- en el espacio latente. La red decodificadora, también llamada *red generativa*, transforma la distribución de probabilidad del espacio latente en una distribución de probabilidad en el espacio de características.

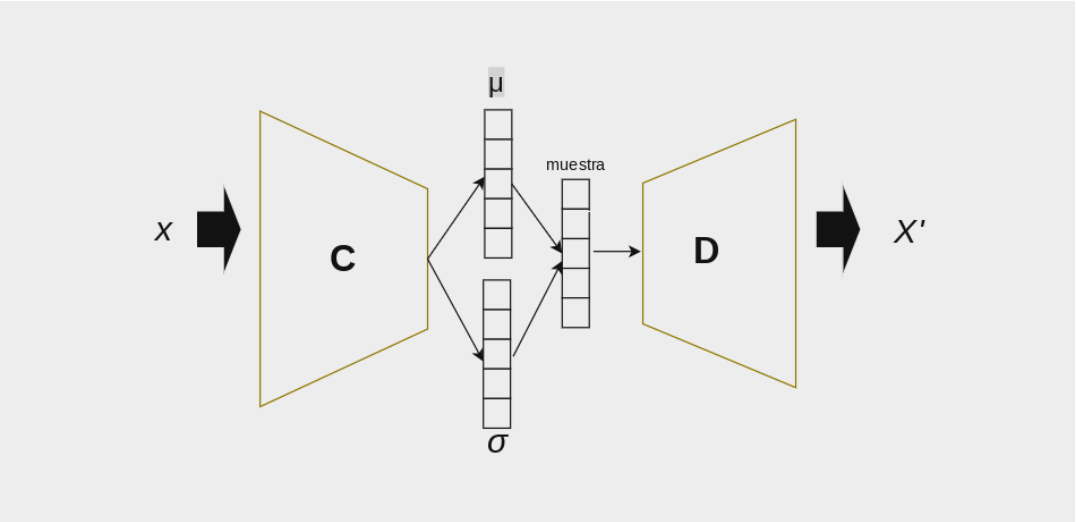


FIGURE 3.3: Autocoficadores Variacionales

Dado un conjunto de datos de entrada $x = \{x_1, x_2, ..., x_N\}$, donde $x_i \in \mathbb{R}^m$, se asume que cada muestra es generada por un mismo proceso o sistema subyacente cuya distribución de probabilidad se desconoce. El modelo buscado procura aprender $p_\theta(x)$, donde θ son los parámetros de la función. Por las ventajas que ofrece el logaritmo[1] para el cálculo de la misma tendremos la siguiente expresión:

$$\log p_\theta(x) = \sum_{x_i \in x} \log p_\theta(x)^{[2]}$$

La forma más común de calcular el parámetro θ es a través del estimador de *máxima verosimilitud*, cuya función de optimización es: $\theta^* = \arg\max_\theta \log p_\theta(x)$, es decir, buscamos los parámetros θ que maximizan la log-verosimilitud asignada a los datos por el modelo.

En el contexto de los AVs, el objetivo es modelar la distribución de probabilidad de los datos observados x a través de una distribución de probabilidad conjunta de variables observadas y latentes: $p_\theta(x, z)$. Aplicando la regla de la cadena de probabilidad podemos factorizar la distribución conjunta de la siguiente manera: $p_\theta(x, z) = p_\theta(x|z)p_\theta(z)$. Aquí $p_\theta(x|z)$ es la probabilidad condicional de los datos observados dados los latentes, y $p_\theta(z)$ es la probabilidad *a priori*[3] de los latentes.

Para determinar la distribución marginal respecto de los datos observados, es preciso integrar sobre todos los elementos de z, dando como resultado la siguiente función:

[1] El logaritmo convierte la probabilidad conjunta (que se calcula como el producto de las probabilidades condicionales) en una suma de logaritmos, facilitando el cálculo y evitando problemas de precisión numérica: $\log(ab) = \log(a) + \log(b)$.

[2] Esta función se lee como la log-verosimilitud de los datos observados x bajo el modelo $p_\theta(x)$ y es igual a la suma de la log-verosimilitud de cada dato de entrada x_i.

[3] La expresión *a priori* alude a que no está condicionada por ningun dato observado.

$p_\theta(x) = \int p_\theta(x, z)dz$ [4]

Esta distribución marginal puede ser extremadamente compleja, y contener un número indeterminable de dependencias (Kingma and Welling 2019), volviendo el calculo de la verosimilitud de los datos observados intratable. Esta intratabilidad de $p_\theta(x)$ está determinada por la intratabilidad de la distribución *a posteriori* $p_\theta(z|x)$, cuya dimensionalidad y multi-modalidad pueden hacer difícil cualquier solución analítica o numérica eficiente. Dicho obstáculo impide la diferenciación y por lo tanto la optimización de los parámetros del modelo.

Para abordar este problema, se acude a la inferencia variacional que introduce una aproximación $q_\phi(z|x)$ a la verdadera distribución *a posteriori* $p_\theta(z|x)$. Generalmente se emplea la distribución normal multivariada para aproximar la distribución *a posteriori*, con media y varianza parametrizadas por la red neuronal[5]. Sin embargo, la elección de la distribución no necesariamente tiene que pasar por una distribución normal, el único requerimiento es que sea una distribución que permita la diferenciación y el cálculo de la divergencia entre ambas distribuciones (por ejemplo si X es binaria la distribución $p_\theta(x|z)$ puede ser una distribución Bernoulli).

Así, en lugar de maximizar directamente el logaritmo de la verosimilitud (*log-verosimilitud*), se maximiza una cota inferior conocida como *límite inferior de evidencia* (*ELBO* por sus siglas en ingles). La derivación procede de la siguiente manera:

1. Log-verosimilitud marginal:

   $\log p_\theta(x) = \log \left(\int p_\theta(x, z)\, dz \right)$

2. Aplicando inferencia variacional:

   $\log p_\theta(x) = \log \left(\int q_\phi(z|x) \frac{p_\theta(x,z)}{q_\phi(z|x)}\, dz \right)$

3. Aplicando la desigualdad de Jensen[6]:

   $\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x,z)}{q_\phi(z|x)} \right) \right]$

4. Descomponiendo la fracción dentro del logaritmo:

   $\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z) + \log p_\theta(z) - \log q_\phi(z|x) \right]$

5. El resultando es el límite inferior de evidencia:

   $\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\mathrm{KL}}(q_\phi(z|x) \| p_\theta(z))$

Donde:

[4] Aquí dz es el diferencial de z, por lo que la expresión indica la integración sobre todas las posibles configuraciones de la variable latente.

[5] En AVs, se suele asumir que z sigue una distribución normal multivariada: $p_\theta(z) = \mathcal{N}(z; 0, I)$, con media cero y matriz de covarianza identidad. La matriz de covarianza identidad es una matriz diagonal con unos en la diagonal y ceros en los demás lugares, y su empleo simplifica la implementación del modelo, permite que las variables latentes sean independientes (covarianza = 0) y varianza unitaria, evitando así cualquier complejidad vinculada a las dependencias entre dimensiones de z.

[6] Nótese que ese límite es siempre menor o igual y esto se deriva de una de las propiedades de las funciones convexas. Esta propiedad, denominada *desigualdad de Jensen*, establece que el valor esperado de una función convexa es siempre mayor o igual a la función del valor esperado. Es decir, $\mathbb{E}[f(x)] \geq f(\mathbb{E}[x])$. En el caso de funciones cóncavas, la desigualdad se invierte: $\mathbb{E}[f(x)] \leq f(\mathbb{E}[x])$. En este caso, la función logaritmo es cóncava, por lo que la desigualdad se expresa como: $\log(\mathbb{E}[x]) \geq \mathbb{E}[\log(x)]$.

- $\mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$ es el valor esperado (*esperanza*[7]) de la log-verosimilitud bajo la aproximación variacional, y determina la precision de la reconstrucción de los datos de entrada (un valor alto de esta esperanza indica que el modelo es capaz de reconstruir los datos de entrada con alta precisión a partir de los parámetros generados por $q_\phi(z|x)$).
- $D_{\mathrm{KL}}(q_\phi(z|x)\|p_\theta(z))$ es la divergencia de Kullback-Leibler entre la distribución $q_\phi(z|x)$ y la distribución *a priori* de las variables latentes $p_\theta(z)$, y determina la regularización del espacio latente.

Maximizando esta cota inferior (*ELBO*), se optimizan simultáneamente los parámetros θ del modelo y los parámetros ϕ de la distribución empleada en la aproximación, permitiendo una inferencia eficiente y escalable en modelos con z de alta dimensionalidad [8].

El objetivo de aprendizaje del AV se da entonces por:

$$\mathcal{L}_{\theta,\phi}(x) = \max(\phi,\theta)\left(E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\mathrm{KL}}(q_\phi(z|x)\|p_\theta(z))\right),$$

Como puede apreciearse en la ecuación anterior la función de pérdida del AV se compone de dos términos: el primero es la esperanza de la log-verosimilitud bajo la aproximación variacional y el segundo es la divergencia de Kullback-Leibler relacionada a la reconstrucción de los datos y la regularización del espacio latente. Existe entre ambos términos una relación de compromiso que permite al AV aprender una representación representativa de los datos de entrada y, al mismo tiempo, un espacio latente continuo y regularizado. Cuanto mayor sea la divergencia de Kullback-Leibler, más regularizado será el espacio latente y más suave será la distribución de probabilidad de los datos generados. Cuanto menor sea la divergencia de Kullback-Leibler, más se parecerá la distribución de probabilidad de los datos generados a la distribución de probabilidad de los datos de entrada, sin embargo, el espacio latente será menos regularizado y la generación de datos mas ruidosa.

[7] La esperanza es un promedio ponderado de todos los posibles valores que puede tomar una variable aleatoria, donde los pesos son las probabilidades de esos valores. En un AV, donde consideramos una distribución aproximada $q_\phi(z|x)$ para el espacio latente, la expresión citada es la esperanza de la log-verosimilitud bajo esta distribución.Aunque teóricamente esto implica un promedio sobre todas las posibles muestras z de la distribución $q_\phi(z|x)$, en la práctica, esta esperanza se estima utilizando una única muestra durante el entrenamiento por razones de eficiencia computacional. Esta única muestra permite calcular directamente $\log p_\theta(x_i|z_i)$, proporcionando una aproximación a la esperanza teórica y determinando la precisión de la reconstrucción de los datos de entrada.

[8] En la teoría, cuando derivamos el objetivo de un VAE, estamos maximizando el límite inferior variacional (ELBO), para que la aproximación sea lo más cercana posible a la verdadera distribución de los datos. LLevado el problema a una implementación práctica generalmente se emplean optimizadores que minimizan una función de pérdida. Para convertir el problema de maximización del ELBO en un problema de minimización, simplemente negamos el ELBO, y reescribimos los términos de la ecuación (tando el error de reconstrucción como la divergencia entre distribuciones) para trabajar con sumas de cantidades positivas. Esto se debe a que, en general, los optimizadores (SGD, Adam, etc.) están diseñados para minimizar funciones de pérdida.

Capítulo 4

Algoritmos Genéticos

Los algoritmos genéticos (en adelante AG) son métodos de optimización inspirados en la evolución natural, diseñados para encontrar soluciones en espacios de búsqueda complejos (Vignolo and Gerard 2017). A diferencia de los métodos de optimización exhaustivos (ej. métodos enumerativos[1]), los AG son particularmente efectivos en espacios de búsqueda discretos, ruidosos, cuando la función objetivo no puede describirse mediante una ecuación o la misma no es diferenciable (Goldberg, David E. 1989). Utilizando principios basados en la evolución, estos algoritmos generan iterativamente soluciones a partir de una población de candidatos, de manera similar a cómo la evolución natural optimiza características biológicas a lo largo de generaciones en función de las condiciones del entorno. En contextos de aplicación sus resultados regularmente conducen a soluciones cercanas al óptimo, capaces de mantener un buen compromiso en la satisfacción de múltiples requerimientos (Jiao et al. 2023). Por eso, los AG son eficaces para atacar tanto problemas de objetivo único, como problemas multiobjetivo.

La robustez de los AG está determinada, como bien sostiene Goldberg (1989), por una serie de características distintivas, que fortalecen su configuración de búsqueda, a saber: a) operan sobre un espacio *codificado* del problema y no sobre el espacio en su representación original; b) realizan la exploración evaluando una *población de soluciones* y no soluciones individuales; c) tienen como guía una *función objetivo* (también llamada *función de aptitud*) que no requiere derivación u otras funciones de cálculo; y d) suponen *métodos probabilísticos de transición* (operadores estocásticos) y no reglas determinísticas. Estas características permiten a los AG superar restricciones que tienen otros métodos de optimización, condicionados -por ejemplo- a espacios de búsqueda continuos, diferenciables o unimodales. Por ello, su aplicación se ha difundido notablemente, trascendiendo los problemas clásicos de optimización, aplicándose en distintas tareas (Vie, Kleinnijenhuis, and Farmer 2021) y a lo largo de diversas industrias (Jiao et al. 2023).

La importancia de los AGs como herramientas de optimización, adquiere especial preeminencia en el problema de *selección de características* (Jiao et al. 2023), por lo que en este trabajo dirigiremos la atención en esa dirección. La *selección de características* (en adelante SC) representa un desafío de optimización combinatoria complejo, que despierta interés en el universo del aprendizaje automático debido a su impacto en el rendimiento de los modelos y la posibilidad de reducir la complejidad computacional de ciertos problemas. Tal desafío está determinado por varios factores. En primer lugar encontramos que, en espacios de alta dimensionalidad, la cardinalidad del conjunto de soluciones candidatas crece de manera exponencial, y los problemas

[1] Goldberg, David E. (1989), p.4.

se vuelven computacionalmente intratables debido a la extensión del espacio de búsqueda.[2] En segundo lugar, junto con la alta dimensionalidad, aparece el problema de las interacciones entre características. Aquí, el prolífico espectro de dependencias que pueden establecer los atributos plantea normalmente vínculos difíciles de modelar atento a que se multiplican de la mano de la dimensionalidad.[3] Por último, aunque no por ello menos importante, aparece el carácter multiobjetivo de los problema de SC, donde no solo interesa maximizar la eficacia de los modelos sino también que sean eficientes. Eficiencia que implica -generalmente- la necesidad de minimizar la cantidad de atributos seleccionados para resolver un problema (Jiao et al. 2023).

Estos desafíos son abordados por los AGs de manera conveniente y creativa.[4] En el marco de este algoritmo cada individuo (muestra) representa una solución candidata, con un perfil genético particular determinado por un subconjunto de características. La búsqueda de las mejores soluciones comienza con la selección de una población inicial de individuos y un subconjunto de características generados aleatoriamente. Este subconjunto se evalúa utilizando una función de aptitud, y los individuos con mejor rendimiento (puntaje) son seleccionados para la reproducción. Este proceso continúa durante un cierto número de generaciones hasta que se cumple una condición de terminación (Goldberg, David E. 1989).

Este mecanismo simple constituye un eficaz método de selección en contextos de alta dimensionalidad y bajo número de muestras. Esa eficacia se debe a la capacidad de explorar el problema dividiéndolo en subespacios de características y, al mismo tiempo, explotar las regiones de mayor valor en cada subespacio (Goldberg, David E. 1989).[5]

Dicho lo anterior, no es menos cierto que la capacidad de selección de los AGs depende de la evaluación de aptitud que orienta la búsqueda de las mejores soluciones, y tal evaluación descansa -finalmente- en la disponibilidad de datos. En efecto, la existencia y número de individuos condiciona la función objetivo y por esa vía también al proceso de selección de características de los AGs. La disponibilidad de datos resulta así un factor clave para la selección. Este requerimiento, vinculado particularmente a la función objetivo, se presenta no solo cuando se utiliza como evaluador a modelos complejo de aprendizaje automático (que demandan una cantidad creciente de muestras de entrenamiento)[6], sino también cuando se trabaja sobre datos cuyas clases se

[2] Cabe destacar que para un conjunto de `n` características es posible determinar un total de `n2` posibles soluciones, espacio que constituye un dominio de búsqueda difícil de cubrir aún con `n` conservadores. Por ejemplo para un conjunto de 20 características (atributos) el número total de subconjuntos a evaluar supera el millón de posibles candidatos, específicamente: 1.048.576.

[3] Por ejemplo, dos características con alto valor discriminatorio para resolver un problema de clasificación pueden ser redundantes debido a su correlación y exigir criterios inteligentes de inclusión-exclusión. A la inversa, características que individualmente consideradas pueden carecer de valor discriminatorio, debido a su complementariedad pueden ser esenciales para resolver un problema y por lo tanto exigir criterios complejos de evaluación y búsqueda.

[4] Ciertamente, no son sus atributos aislados los que le dan esa posibilidad, sino la interacción de sus componentes.

[5] Ambas funciones -exploración y explotación- permiten al algoritmo reconfigurar el espacio de búsqueda y poner a prueba sus complejas dependencias. Como vimos, el procedimiento es orientado por una función de aptitud que evalúa las distintas posibilidades combinatorias encontradas por el algoritmo y retroalimenta el proceso exploratorio. La dinámica completa tiene como resultado un procedimiento experimental de búsqueda y selección capaz de reconocer soluciones próximas al óptimo.

[6]

1.

encuentran desbalanceadas (Fajardo et al. 2021; Blagus and Lusa 2013). En ambos escenarios, la falta de información suficiente degrada la capacidad informativa de la función objetivo (Hastie, Tibshirani, and Friedman 2009), afectando gravemente el proceso de selección de características.

En esa línea, el problema de la disponibilidad de datos en los proyecto de selección de características -sea dentro o fuera del campo de los AGs-, ha encontrado en las estrategias de aumentación una posible solución (Gm et al. 2020). Entre esas estrategias, los Autoencoders Variacionales (en adelante AV) han adquirido popularidad, superando a métodos tradicionales (ej. sobremuestreo (Blagus and Lusa 2013)) y -en ciertos casos- también a otro modelos generativos basados de redes neuronales profundas (Fajardo et al. 2021).

Los AVs constituyen modelos generativos[7] capaces de aprender una representación latente de datos observados y producir nuevas muestras con las mismas características fundamentales[8] que las observaciones (Kingma and Welling 2019). Esa capacidad resulta particularmente efectiva por el hecho de que prescinde de fuertes supuestos estadísticos a los que adscriben otros modelos generativos y también por su escalabilidad.[9] Hoy los AVs son ampliamente utilizados en biología molecular, química, procesamiento de lenguaje natural, astronomía, entre otros (Ramchandran et al. 2022).

Por todo lo visto hasta aquí advertimos que la posibilidad de expandir el conjunto de datos mediante el uso de AVs abre nuevas alternativas para afrontar el problema de la selección de características aplicando AGs. Estas alternativas no solo parecen prometedoras como estrategias orientadas a la multiplicación de muestras de entrenamiento para mejorar el desempeño de la función objetivo, sino también como partes funcionales de sus operadores de variación.[^10] De este modo, la integración de ambas tecnologías ofrece un enfoque provechoso para abordar el problema de selección de características en distintos escenarios que enfrentan los AGs.

4.1. AG version 2

Para el presente trabajo usaremos algoritmos genéticos (AGs) como método de búsqueda[10] debido a la posibilidad que brindan de emplear codificación binaria y permitir así una representación intuitiva del espacio de características (Vignolo and Gerard 2017). Para aumentación de datos utilizaremos *autoencoders variacionales* (AVs) como instancia generativa (Kingma and Welling 2019).

Los AGs constituyen una de las herramientas más estudiadas e implementadas dentro de los métodos evolutivos Kramer (2017), dada su capacidad para encontrar soluciones en espacios de búsqueda complejos (Vignolo and Gerard 2017). El procedimiento de búsqueda de los AGs opera evolucionando una población de individuos que consisten en cromosomas que codifican el espacio de soluciones. Dicha evolución -al igual que la evolución natural- sucede a través de operadores (funciones) de selección, variación (mutación y cruce) y reemplazo que transforman el material genético disponible:

[7] Redes neuronales profundas con arquitectura *encoder-decoder* (Kingma and Welling 2019). Estos modelos pueden presentar distintas configuraciones según el problema tratado y el objetivo particular de la implementación (Wu, Cao, and Qi 2023).

[8] Similar distribución conjunta de probabilidad.

[9] El modelo emplea *retropropagación* como estrategia de optimización (Kingma and Welling 2019)

[10] Otros métodos robustos, como por ejemplo el *enjambre de partículas* (PSO) y *optimización de colonia de hormigas* (ACO), típicamente utilizan codificación basada en números reales por lo que constituyen opciones menos adecuadas al problema que enfrentaremos en este trabajo.

los individuos más aptos sobreviven y se reproducen, mientras que los menos aptos desaparecen[11]. Esta aptitud -que imita la presión selectiva de un entorno natural- se evalúa mediante la aplicación de una función objetivo (específica del problema) a cada individuo a partir de la información decodificada de sus cromosomas. Dicha función objetivo puede asumir múltiples formas (Jiao et al. 2023), pero en nuestro trabajo nos centraremos en el uso de modelos de aprendizaje automático, particularmente Maquinas de Soporte Vectorial (Boser, Guyon, and Vapnik 1992) y Bosques Aleatorios (Breiman 2001). Este método heurístico de búsqueda tendrá en nuestro trabajo dos configuraciones: una *clásica* sin aumentación de datos y una *novedosa* con aumentación de datos aplicando *autoencoders variacionales* (AV).

[11] Como su nombre lo indica el operador de selección determina la elegibilidad de un individuo para sobrevivir y reproducirse en función de su aptitud para resolver un problema. En el contexto de los AGs esta aptitud no es otra cosa que el puntaje que obtiene un individuo evaluado en una función objetivo. Por su parte los operadores de variación tienen como función combinar la información genética de individuos (cruce) y alterar aleatoriamente sus cromosomas (mutación), promoviendo transformaciones en el material genético global con sesgo hacia mejorar la aptitud poblacional para resolver un problema. La variación equivale a la búsqueda natural por mejorar las adaptaciones de los individuos a su entorno. Finalmente el operador de reemplazo mantiene la población constante, sustituyendo individuos poco aptos por aquellos de mayor aptitud. Estos operadores se combinan en ciclos iterativos que se repiten hasta satisfacer un criterio de terminación deseado (por ejemplo, un número predefinido de generaciones o un valor de aptitud) (Vignolo and Gerard 2017).

Capítulo 5

Algo

Aquí digo algo importante.

Capítulo 6

Algo

Aquí digo algo importante.

References

Blagus, Rok, and Lara Lusa. 2013. "SMOTE for High-Dimensional Class-Imbalanced Data." *BMC Bioinformatics* 14 (1): 106. https://doi.org/10.1186/1471-2105-14-106.

Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. "A Training Algorithm for Optimal Margin Classifiers." In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144–52. COLT '92. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/130385.130401.

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32. https://doi.org/10.1023/A:1010933404324.

Doersch, Carl. 2021. "Tutorial on Variational Autoencoders." January 3, 2021. http://arxiv.org/abs/1606.05908.

Fajardo, Val Andrei, David Findlay, Charu Jaiswal, Xinshang Yin, Roshanak Houmanfar, Honglei Xie, Jiaxi Liang, Xichen She, and D. B. Emerson. 2021. "On Oversampling Imbalanced Data with Deep Conditional Generative Models." *Expert Systems with Applications* 169 (May): 114463. https://doi.org/10.1016/j.eswa.2020.114463.

Gm, Harshvardhan, Mahendra Kumar Gourisaria, Manjusha Pandey, and Siddharth Swarup Rautaray. 2020. "A Comprehensive Survey and Analysis of Generative Models in Machine Learning." *Computer Science Review* 38 (November): 100285. https://doi.org/10.1016/j.cosrev.2020.100285.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York, NY, USA: Addison-Wesley.

Golub, T. R., D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, et al. 1999. "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring." *Science* 286 (5439): 531–37. https://doi.org/10.1126/science.286.5439.531.

Hastie, T, R Tibshirani, and J Friedman. 2009. *The Element of Statistical Learning: Data Mining, Inference, and Prediction*. Second Edition. Springer.

Isabelle Guyon, Steve Gunn. 2004. "Gisette." UCI Machine Learning Repository. https://doi.org/10.24432/C5HP5B.

Jiao, Ruwang, Bach Hoai Nguyen, Bing Xue, and Mengjie Zhang. 2023. "A Survey on Evolutionary Multiobjective Feature Selection in Classification: Approaches, Applications, and Challenges." *IEEE Transactions on Evolutionary Computation*, 1–1. https://doi.org/10.1109/TEVC.2023.3292527.

Kingma, Diederik P., and Max Welling. 2019. "An Introduction to Variational Autoencoders." *Foundations and Trends® in Machine Learning* 12 (4): 307–92. https://doi.org/10.1561/2200000056.

Kramer, Oliver. 2017. *Genetic Algorithm Essentials*. Vol. 679. Studies in Computational Intelligence. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-52156-5.

Ramaswamy, Sridhar, Pablo Tamayo, Ryan Rifkin, Sayan Mukherjee, Chen-Hsiang Yeang, Michael Angelo, Christine Ladd, et al. 2001. "Multiclass Cancer Diagnosis

Using Tumor Gene Expression Signatures." *Proceedings of the National Academy of Sciences* 98 (26): 15149–54. https://doi.org/10.1073/pnas.211566398.

Ramchandran, Siddharth, Gleb Tikhonov, Otto Lönnroth, Pekka Tiikkainen, and Harri Lähdesmäki. 2022. "Learning Conditional Variational Autoencoders with Missing Covariates." March 2, 2022. http://arxiv.org/abs/2203.01218.

Torre, Jordi de la. 2023. "Autocodificadores Variacionales (VAE) Fundamentos Te\'oricos y Aplicaciones." February 18, 2023. https://doi.org/10.48550/arXiv.2302.09363.

Vie, Aymeric, Alissa M. Kleinnijenhuis, and Doyne J. Farmer. 2021. "Qualities, Challenges and Future of Genetic Algorithms: A Literature Review." September 13, 2021. http://arxiv.org/abs/2011.05277.

Vignolo, Leandro D., and Matias F. Gerard. 2017. "Evolutionary Local Improvement on Genetic Algorithms for Feature Selection." In *2017 XLIII Latin American Computer Conference (CLEI)*, 1–8. Cordoba: IEEE. https://doi.org/10.1109/CLEI.2017.8226467.

Wu, Zhangkai, Longbing Cao, and Lei Qi. 2023. "eVAE: Evolutionary Variational Autoencoder." January 1, 2023. https://doi.org/10.48550/arXiv.2301.00011.

Apéndice A

Frequently Asked Questions

A.1. How do I change the colors of links?

Pass in `urlcolor:` in yaml. Or set these in the include-in-header file.

If you want to completely hide the links, you can use:

{`\hypersetup{allcolors=.}`}, or even better:

{`\hypersetup{hidelinks}`}.

If you want to have obvious links in the PDF but not the printed text, use:

{`\hypersetup{colorlinks=false}`}.