University of Ottawa

Doctoral Thesis

A doctoral thesis title

Author:
Jane Doe

Supervisor:
Dr. Ashok Kunil

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Informatics Program
Department of Mathematics

December 15, 2023

Declaration of Authorship

I, Jane Doe, declare that this thesis titled, A doctoral thesis titleand the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"*Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.*"

Dave Barry

UNIVERSITY OF OTTAWA

Resumen

Applied Math Group

Department of Mathematics

Doctor of Philosophy

A doctoral thesis title

by Jane Doe

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam quis accumsan ante. Quisque lorem metus, varius id urna eget, lacinia dapibus sem. Etiam laoreet, quam ac mollis congue, arcu leo dictum neque, nec euismod sem enim luctus odio. Donec condimentum tortor sit amet mollis volutpat. Donec ornare libero vel velit malesuada consectetur. Vestibulum in sem non justo dignissim congue at quis erat. Integer quis erat vitae mi maximus fringilla tristique nec odio. Morbi non ipsum sapien. Vestibulum tortor est, ultricies in eros et, bibendum iaculis justo. Cras pellentesque enim quam, id pretium lacus lacinia non. Integer velit neque, ultrices a malesuada vel, imperdiet quis enim. Quisque eu facilisis urna, ut faucibus lorem. Donec mollis turpis sed arcu venenatis interdum. Nulla facilisis tortor ac scelerisque consequat.

Acknowledgements

Integer id risus vel lorem laoreet commodo lobortis quis purus. Cras cursus leo vel dui laoreet pulvinar. Nunc tincidunt metus et ante fermentum lacinia. Proin quam magna, tristique ut viverra at, dapibus eget elit. Quisque eu leo id nisi semper laoreet at ac nulla. Fusce volutpat, metus sed dictum mattis, nisl elit dapibus velit, non porttitor urna urna vel diam. Praesent tortor nulla, rutrum ac magna a, tempor sagittis enim. Praesent pharetra ipsum libero, eu malesuada libero blandit ut. Sed sed venenatis ligula, nec convallis turpis. Nulla iaculis felis eros, eget pharetra lorem cursus quis. Nunc iaculis lobortis magna at malesuada. Nullam elementum elit at urna congue aliquam.

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

For Elsa

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 (**isabelleguyonGisette2004?**). 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 (**golubMolecularClassificationCancer1999?**) 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 (**ramaswamyMulticlassCance**
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 establecido una búsqueda en grilla de sus respectivos valores. Hemos establecido 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 fragmento de código:

```
parameters = {
    "LDA": {
        "classifier__solver": ["svd"]
    },
    "QDA": {
        "classifier__reg_param": [0.01 * ii for ii in range(0, 101)]
    },
    "AdaBoost": {
        "classifier__estimator": [DecisionTreeClassifier(max_depth=ii) for ii in ra
        "classifier__n_estimators": [200],
        "classifier__learning_rate": [0.001, 0.01, 0.05, 0.1, 0.25, 0.50, 0.75, 1.0
    },
    "Bagging": {
        "classifier__estimator": [DecisionTreeClassifier(max_depth=ii) for ii in ra
        "classifier__n_estimators": [200],
        "classifier__max_features": [0.2, 0.4, 0.6, 0.8, 0.9, 1.0],
        "classifier__n_jobs": [-1]
    },
    # Update dict with Gradient Boosting
    "Gradient Boosting": {
                            "classifier__learning_rate":[0.15,0.1,0.01,0.001],
                            "classifier__n_estimators": [200],
                            "classifier__max_depth": [2,3,4,5,6],
                            "classifier__min_samples_split": [0.005, 0.01, 0.05, 0.
                            "classifier__min_samples_leaf": [0.005, 0.01, 0.05, 0.1
                            "classifier__max_features": [ "sqrt", "log2"],
                            "classifier__subsample": [0.8, 0.9, 1]
    },
    # Update dict with Extra Trees
    "Extra Trees Ensemble": {
                            "classifier__n_estimators": [200],
                            "classifier__class_weight": [None, "balanced"],
                            "classifier__max_features": [ "sqrt", "log2"],
                            "classifier__max_depth" : [3, 4, 5, 6, 7, 8],
                            "classifier__min_samples_split": [0.005, 0.01, 0.05, 0.
```

```
                        "classifier__min_samples_leaf": [0.005, 0.01, 0.05, 0.10],
                        "classifier__criterion" :["gini", "entropy"],
                        "classifier__n_jobs": [-1]
    },
    # Update dict with Random Forest Parameters
    "Random Forest": {
                        "classifier__n_estimators": [200],
                        "classifier__class_weight": [None, "balanced"],
                        "classifier__max_features": [ "sqrt", "log2"],
                        "classifier__max_depth" : [3, 4, 5, 6, 7, 8],
                        "classifier__min_samples_split": [0.005, 0.01, 0.05, 0.10],
                        "classifier__min_samples_leaf": [0.005, 0.01, 0.05, 0.10],
                        "classifier__criterion" :["gini", "entropy"]     ,
                        "classifier__n_jobs": [-1]
    },
    # Update dict with Ridge
    "Ridge": {
                "classifier__alpha": [1e-7, 1e-5, 1e-3, 1e-2, 1e-1, 0.25, 0.50, 0.75, 1
    },
    # Update dict with SGD Classifier
    "SGD": {
            "classifier__alpha": [1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 0.25, 0.50,
            "classifier__penalty": ["l1", "l2"],
            "classifier__n_jobs": [-1]
    },
    # Update dict with BernoulliNB Classifier
    "BNB": {
            "classifier__alpha": [1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 0.25, 0.50,
    },
    # Update dict with GaussianNB Classifier
    "GNB": {
            "classifier__var_smoothing": [1e-9, 1e-8,1e-7, 1e-6, 1e-5]
    },
    # Update dict with K Nearest Neighbors Classifier
    "KNN": {
            "classifier__n_neighbors": list(range(1,31)),
            "classifier__p": [1, 2, 3, 4, 5],
            "classifier__leaf_size": [5, 10, 20, 30, 40, 50],
            "classifier__n_jobs": [-1]
    },
    # Update dict with MLPClassifier
    "MLP": {
            "classifier__hidden_layer_sizes": [(5,5), (10,10), (5,5,5), (10,10,10)],
            "classifier__activation": ["identity", "logistic", "tanh", "relu"],
            "classifier__learning_rate": ["constant", "invscaling", "adaptive"],
            "classifier__max_iter": [500, 1000, 2000],
            "classifier__alpha": list(10.0 ** -np.arange(1, 10)),
    },
    "LSVC": {
```

```
            "classifier__penalty": ["l2"],
            "classifier__C": [0.0001, 0.01, 0.1, 1.0, 10, 100]
    },
    "NuSVC": {
            "classifier__nu": [0.25, 0.50, 0.75],
            "classifier__kernel": ["linear", "rbf", "poly"],
            "classifier__degree": [1,3,5,6],
    },
    "SVC": {
            "classifier__kernel": ["linear", "rbf", "poly"],
            "classifier__gamma": ["auto"],
            "classifier__C": [0.1, 0.5, 1, 10, 50, 100],
            "classifier__degree": [1, 3, 5, 6]
    },
    # Update dict with Decision Tree Classifier
    "DTC": {
            "classifier__criterion" :["gini", "entropy"],
            "classifier__splitter": ["best", "random"],
            "classifier__class_weight": [None, "balanced"],
            "classifier__max_features": [ "sqrt", "log2"],
            "classifier__max_depth" : [1,2,3, 4, 5, 6, 7, 8],
            "classifier__min_samples_split": [0.005, 0.01, 0.05, 0.10],
            "classifier__min_samples_leaf": [0.005, 0.01, 0.05, 0.10],
    },
    # randomized decision trees: a.k.a. extra-trees
    "ETC": {
        "classifier__criterion": ["gini", "entropy"],
        "classifier__splitter": ["best", "random"],
        "classifier__class_weight": [None, "balanced"],
        "classifier__max_features": ["sqrt", "log2"],
        "classifier__max_depth": [1, 3, 5, 7, 8],
        "classifier__min_samples_split": [0.005, 0.01, 0.05, 0.10],
        "classifier__min_samples_leaf": [0.005, 0.01, 0.05, 0.10]
    }
}
```

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	-	-	-	-

Models	Leukemia Train	Leukemia Test	Madelon Train	Madelon Test	Gisette Train	Gisette Test	GCM Train	GCM Test
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:

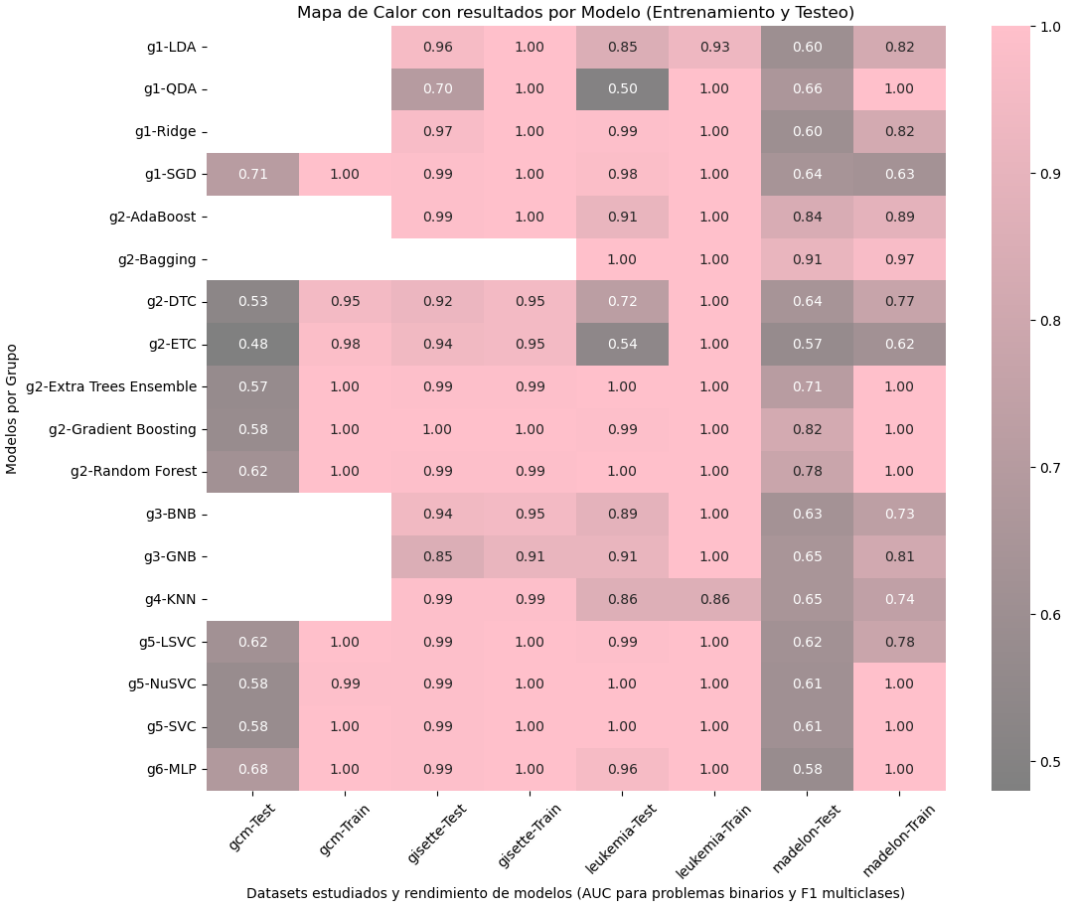

FIGURE 2.1: algoritmosclasicos

Capítulo 3

Autoencoder Variacional

En este capítulos presentaremos la arquitectura de nuestro modelo de Autoencoder Variacional (VAE). Expondremos brevemente los pasos seguidos en su construcción y los resultados obtenidos en distintas evaluaciones de desempeño.

3.1. Presentación del modelo

Capítulo 4

Algo

Aquí digo algo importante.

Capítulo 5

Algo

Aquí digo algo importante.

Capítulo 6

Algo

Aquí digo algo importante.

References

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}`}.