

Universidad de La Habana
Facultad de Matemática y Computación



Optimización Multiobjetivo para Autogoal

Autor:

Rodrigo Daniel Pino Trueba

Tutores:

Suilan Estevez

Daniel Valdes

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

Fecha

github.com/username/repo

Dedicación

Agradecimientos

Agradecimientos

Opinión del tutor

Opiniones de los tutores

Resumen

Resumen en español

Abstract

Resumen en inglés

Índice general

Introducción	1
1. Estado del Arte	4
1.1. Optimización Multiobjetivo	4
1.1.1. Técnicas de Escalarización (Scalarization)	5
1.1.2. Algoritmos Numéricos	6
1.1.3. Algoritmos Genéticos Multiobjetivos (MOEA)	6
1.2. Aprendizaje de Máquina Automatizado	7
1.3. Optimización Multiobjetivo en AutoML	8
2. Propuesta	9
3. Detalles de Implementación y Experimentos	10
Conclusiones	11
Recomendaciones	12
Bibliografía	13

Índice de figuras

Ejemplos de código

Introducción

Aprendizaje de Máquina (ML por sus siglas en inglés) es una rama de la Inteligencia Artificial enfocada en entender y construir programas que “aprendan” y logren buenos resultados en un conjunto de tareas definidas (Mitchell y col. 1990). Este campo que vio sus inicios sobre la década de los 50 no tuvo auge hasta la primera década del nuevo milenio gracias al auge de las nuevas tecnologías en el campo de la computación. Se comenzó a utilizar ampliamente en diferentes aplicaciones donde definir algoritmos convencionales era muy difícil.

Construir un flujo válido de Aprendizaje de Máquina requiere tener al menos un experto en el tema pero estos son escasos lo que hace la herramienta parcialmente inaccesible. Se crea un nuevo campo que estudia el proceso de automatizar la creación de estos flujos de ML (conocido por AutoML por sus siglas en inglés). Esto impulsa aún más la adopción del aprendizaje de máquina ya que se vuelve accesible a todos.

El AutoML funciona dado un set de datos y una métrica busca el mejor algoritmo de ML de acuerdo a la puntuación de dicha métrica. Usualmente lo que buscan es exactitud en sus datos.

Con el uso masivo que tiene los algoritmos de Aprendizaje de Máquina en la actualidad, y se forma parte de la vida diaria, ya obtener un sistema ML que tenga resultados relevantes no es el desafío. Se empiezan a querer nuevas features que permitan mayor control sobre el sistema que se genera. Hay sistemas que están dispuestos a sacrificar un poco de relevancia si se puede obtener un flujo de ML menos complejo (y evitar el overfit en el proceso). Otro que quieren tener sistemas “interpretables”, sistemas que permiten a usuarios no expertos entender por que el algoritmo de ML tomó cierta decisión.

La adopción masiva saca a relucir problemas desconocidos como sistemas de ML que producen resultados parcializados dependiendo de ciertas características de estos. Esto tiene importancia especialmente cuando los datos representan seres humanos, y las decisiones del sistema tienen un impacto en el individuo o grupo. No se desea que juzgue por el color de piel o género. Nace la necesidad de producir sistemas justos que consideren ciertos atributos como protegidos y minimicen el sesgo que esto puedan provocar en sus decisiones.

Motivación

Un modelo de Aprendizaje de Máquina o *Machine Learning(ML)* tiene como objetivo maximizar la relevancia de sus resultados dado una métrica. Para medir dichas relevancia, se utilizan métricas como *accuracy* o *f-score*.

Cuando el proposito de un programa de AutoML es producir un algoritmo ML que minimice la relevancia a costa de cualquier otro tipo de criterio se producen buenas predicciones, pero no necesariamente convenientes. Existen casos donde el investigador esta dispuesto a sacrificar un poco de precisión en las predicciones en favor de otras métricas dependiendo del contexto donde se encuentre.

Existen situaciones donde la *interpretabilidad* que implica saber, sin ser un experto en Aprendizaje de Máquina, porque el sistema tomó ciertas decisión puede ser importante. También existen casos donde es necesario producir sistemas de ML justos, para evitar que el algoritmo de ML tenga algún tipo de sesgo por género, color de piel u otro. Estos problemas se resolverían muy bien con multiobjetivo; en caso de no optimizar la relevancia y optimizar únicamente para justeza o interpretabilidad respectivamente produciría un sistema inútil debido a sus malas predicciones.

Optimización multiojetivo en AutoML abrió las puertas también a métricas que no tienen sentido usarlas individualmente como *precisión* y *recobrado*. Precisión mide la proporción entre los valores relevantes identificados y todos los valores identificados mientras que recobrado mide la proporción entre los valores relevantes identificados y todos los valores relevantes. Estas métricas usadas por separadas no son representativas pues basta para tener un recobrado perfecto seleccionar todos los elementos del conjunto de datos, y una precisión casi perfecta seleccionando un pequeño conjunto de elementos. Para lograr un balance entre estas se utiliza *f-score*, que relaciona ambas medidas y permite darle más peso a una o a la otra según determine el investigador, no obstante con OM se pueden obtener un sistema ML que optimice para ambas.

Un framework de *Automated Machine Learning(AutoML)* que pueda optimizar simultáneamente para varios objetivos, se beneficia al ser capaz de ofrecer un conjunto de modelos igual de buenos, y dejar al investigador que seleccione el que más le convenga según su contexto.

Antecedentes

En el entorno del grupo de Inteligencia Artificial de la Universidad de La Habana se desarrolla AutoGOAL una herramienta de AutoML que permite obtener modelos optimizados con respecto a una sola métrica.

Problemática

AutoGOAL no presenta actualmente una herramienta para resolver problemas multiobjetivos. AutoGOAL utiliza Probabilistic Grammatical Evolution (PGE), un algoritmo genético. Implementar un algoritmo de Optimización Multiobjetivo que sea capaz de utilizar la infraestructura ya creada.

Objetivo

Objetivo General

Añadir a AutoGOAL la habilidad de resolver problemas de optimización multiobjetivo.

Objetivos Específicos

- Estudiar el estado del arte relacionado con el problema
- Identificar un algoritmo multiobjetivo que aproveche la estructura de AutoGOAL y su implementación de Probabilistic Grammatical Evolution (PGE)
- Estudiar diferencias cuando se resuelve un mismo problema con optimización multiobjetivo
- Analizar la efectividad de la solución

Estructura de la Tesis

Capítulo 1

Estado del Arte

1.1. Optimización Multiobjetivo

Optimización Multiobjetivo es la rama de la Ciencia y la Matemática que se dedica a optimizar para varias funciones objetivos simultáneamente.

Definición 1.1 *Optimización Multibobjetivo: Dado m funciones objetivos: $f_1 : \mathcal{X} \rightarrow \mathbb{R}, \dots, f_m : \mathcal{X} \rightarrow \mathbb{R}$ que dado un vector en el espacio de decisión \mathcal{X} es transformado en un valor de \mathbb{R} . Más formalmente:*

$$MOP: \min f_1(x), \dots, f_m(x), x \in \mathcal{X}$$

En presencia de varias funciones objetivos ya no es posible hablar de una solución única mejor que el resto. Como las soluciones se comparan respecto a dos o más aspectos, se habla de dominación.

Definición 1.2 *Pareto Dominación: Dados dos vectores en el espacio objetivo, $x \in \mathbb{R}^m$ y $y \in \mathbb{R}^m$, se dice que x Pareto domina a y (i.e. $x \prec y$), si y solo si:*

$$\forall i \in \{1, \dots, m\} : x_i \leq y_i \text{ and } \exists j \in \{1, \dots, m\} : x_j < y_j$$

Existe el caso en el conjunto de soluciones que existan soluciones que no sean dominadas por ninguna otra solución. Estas soluciones vienen siendo el conjunto solución del problema de optimización multiobjetivo.

Definición 1.3 *Frente de Pareto: Todos los vectores x del espacio objetivo \mathcal{Y} tal que no exista $y \prec x$.*

$$\mathcal{P} = \{x | x \in \mathcal{Y}, \neg \exists y \prec x\}$$

Encontrar puntos del frente de Pareto no es tarea difícil, puede ser tan trivial con encontrar el mínimo de cada función objetivo. La complejidad yace en encontrar un conjunto solución que sea lo más aproximado posible al frente de Pareto.

Se debe notar que el termino Optimización Multiobjetivo se utiliza cuando se optimiza simultáneamente para dos o tres funciones objetivos. Para un cantidad de métricas mayor se le conoce coloquialmente en la literatura como Optimización para Muchos Objetivos o *many-objective optimization* en inglés propuesto inicialmente por Fleming y col. 2005. Se hace énfasis en esta diferenciación este pues al aumentar le número de métricas a optimizar:

1. ya no es posible visualizar el frente obtenido;
2. la computación de indicadores o de selección para muchos algoritmos se convierte en problemas NP-duros;
3. existe un rápido crecimiento de puntos no dominados, mientras mayor número de objetivos, la probabilidad de que un punto sea no dominado en un set con distribución normal tiende exponencialmente a 1.

1.1.1. Técnicas de Escalarización (Scalarization)

Los manera clásica de resolver MOP es utilizando técnicas de Escalarización que es cuando de alguna manera u otra se combinan todas las funciones objetivos en una sola. Existen diferentes técnicas de combinar estas funciones:

1. Linear Weighting: Todas las funciones objetivos se combinan en una sola, y a cada una se le asigna un peso. Modificando estos pesos se obtiene una solución distinta del frente de Pareto.

$$\min \sum w_i f_i(x), x \in X$$

Esta enfoque presenta el problema de que cuando el frente de Pareto no es convexo (tiene alguna porción cóncava) no es posible obtener soluciones en esta zona, no importa como se modifiquen los pesos w_i .

2. ϵ -constrain: Se selecciona una función objetivo como principal y las demás se establecen como restricciones de esta, y tienen que ser menor que cierto ϵ_i con $1 \leq n - 1$ por cada función objetivo.

$$\begin{aligned} \min f_1(x), x \in X, \text{ sujeto a:} \\ g_i(x) \leq \epsilon_i \quad \forall i, 0 \leq i \leq n - 1 \end{aligned}$$

Utilizar esta técnica tiene dos dificultades principales, primero la obtención de los ϵ_i , para una adecuada selección requieren conocimiento previo del frente de Pareto y al igual que *Linear Weighting* no es capaz de detectar soluciones en las partes cóncavas del frente.

3. Chebychev Distance (CSP): Se establece un punto de referencia z^* y se utiliza la distancia de Chebychev de los vectores objetivos hacia este como función objetivo utilizando un vector de pesos $\lambda \in \mathbb{R}_{\succ 0}^m$, donde $\mathbb{R}_{\succ 0}^m = \{x | x \in \mathbb{R}^m, x \succ 0\}$.

$$\min \max_{i \in 1, \dots, m} \lambda_i |f_i(x) - z_i^*|, x \in X$$

CSP dado los pesos indicados puede encontrar cualquier punto del frente de Pareto, no importa si es cóncavo o convexo.

1.1.2. Algoritmos Numéricos

En principio todos los algoritmos de escalarización se pueden resolver utilizando métodos numéricos.

Además, existen métodos numéricos que intentan resolver el problema haciendo cumplir las condiciones de Karush-Kuhn-Tucker (Kuhn y Tucker 2014).

La idea va de encontrar al menos una solución al sistema de ecuaciones creado por tratar de resolver KKT. Luego utilizando métodos de continuación y homotopía para añadir al conjunto solución soluciones cercanas a estas. Este algoritmo requiere que las soluciones satisfagan las condiciones de convexidad local y diferenciabilidad, se puede obtener mas información en Hillermeier y col. 2001 y Schütze y col. 2005.

También existen otros métodos para la búsqueda de mínimos globales como *Técnicas de Subdivisión* por Dellnitz y col. 2005, *Optimización Global Bayesiana* por Emmerich, Yang y col. 2016 y *Optimización de Lipschitz* por Žilinskas 2013. Estas requieren de que el espacio de decisión tenga una baja dimensionalidad.

Aun más, también se investiga activamente métodos numéricos que no dependan de derivadas para su resolución utilizando técnicas de búsquedas directas. Ejemplos de algoritmos en Custódio y col. 2011 y Audet y col. 2010.

1.1.3. Algoritmos Genéticos Multiobjetivos (MOEA)

Los algoritmos genéticos tuvieron sus inicios en la década del 60 y fueron usados principalmente en resolución de problemas numéricos combinatorios y no convexos. Utilizan paradigmas extraídos de la naturaleza, tal como selección natural, mutación y recombinación para mover una población (o conjunto de vectores de decisión) hacia una solución óptima (Back 1996).

Los algoritmos genéticos multiobjetivos generalizan esta idea, y son diseñados para en cada iteración acercarse cada vez más al frente de Pareto. Como en este caso no existe solución única la manera de seleccionar los individuos cambia fundamentalmente.

Dentro de los MOEA existen tres paradigmas principales:

1. **MOEA basados en el frente de Pareto:** Se identifican porque dividen el proceso de selección en dos etapas. Una primera donde seleccionan los individuos según su índice de dominación, donde las soluciones que pertenecen al frente de Pareto tienen índice cero. Luego se realiza una segunda selección entre los ya seleccionados utilizando una segunda estrategia de puntuación. NSGA-II (Deb, Pratap y col. 2002) y SPEA2 (Zitzler y Thiele 1999) son algoritmos de este tipo.
2. **Basados en indicador:** Estos utilizan un indicador para calcular cuan cercano es el conjunto actual al frente de Pareto (unario), o cuanto mejora el nuevo conjunto de soluciones respecto a la iteración anterior (binario). Ejemplo de este es SMS-EMOEA (Emmerich, Beume y col. 2005) que suele converger al frente de Pareto con soluciones igualmente distribuidas.
3. **Basados en descomposición:** La idea principal consiste en descomponer el problema en pequeños subproblemas cada uno correspondiente a una sección del frente de Pareto. Por cada sub-problema se resuelve utilizando escalarización con diferente parametrización. El método de escalarización más usado en estos casos suele ser CSP debido a ser capaz de obtener cualquier punto del frente de Pareto. Ejemplo de este paradigma son MOEA/D (Zhang y Li 2007) y NSGA-III (Deb y Jain 2013).

1.2. Aprendizaje de Máquina Automatizado

El Aprendizaje de Máquina Automatizado (Automated Machine Learning, AutoML) consiste en la generación automática de flujos o *pipelines* de Machine Learning que resuelven un problema determinado. El objetivo es transferir el problema de combinación, selección y optimización de hiperparámetros (*Combined Algorithm Selection and Hyperparameter Optimization*, CASH por sus siglas en inglés) del investigador al sistema, permitiéndole a este enfocarse en otras tareas como la validación de los datos o *feature engineering*. También permite al público general y no experto a aplicar ML a sus problemas ya que la parte difícil del trabajo está automatizada.

Existen varias propuestas de Aprendizaje de Máquina Automatizado, utilizando técnicas de variados dominios, como (i) Optimización Bayesiana, (ii) Programación Evolutiva, (iii) Búsqueda Aleatoria, (iv)(v) Aprendizaje de refuerzo y (vi) métodos basados en gradientes.

Específicamente programación evolutiva es de interés para nos

Tree-based Pipeline Optimization Tool (TPOT): Establece un flujo de datos en forma de árbol, donde cada hoja es una copia del dataset, y sobre cada uno se aplican un set de operadores. Se analiza los resultados de aplicar los operadores genetéticos y se seleccionan los k mejores. Luego se aplica regresión logística. Se escogen los mejores 20 pipelines utilizando NSGAI. Donde se optimiza para el mayor accuracy y la menor cantidad de operadores sobre el flujo

RECIPE: programación genética basado en gramáticas probabilísticas

AutoGOAL: basado en gramáticas produce cierta cantidad de pipelines optimizados contra ciertas métricas. Los pipelines se forman utilizando PGE. Se van generando gramática tras gramática hasta que se pase determinando threshold y se devuelve la que mejor optimice contra la métrica dada como input

1.3. Optimización Multiobjetivo en AutoML

Optimización Multiobjetivo está bien establecido en ML como ROC analysis, o biología computacional. Son modelos optimizados para tener un buen performance e interpretabilidad. En el campo de Configuración de Algoritmos introduce multiobjetivo búsqueda iterativa local para configurar SAT solvers, mientras que Zhang introduce basado en carrera?

Aunque es cada vez un feature de más demanda en la actualidad, no se conoce ningún sistema de AutoML que tenga implementado multiobjetivo, excepto por TPOT que optimiza sus pipelines mutuamente para 'accuracy' y tiempo de entrenamiento. No obstante no presenta flexibilidad sobre qué métricas optimizar.

Capítulo 2

Propuesta

Utilizar PGE de Autogoal De donde viene PGE, que es PGE

Utilizar NSGA-II

Que es NSGA-II

Explicar los dos algoritmos de ranking de NSGAII

Como lo que existe ya de AutoGOAL interactua con la nueva implementación de
PGA

Capítulo 3

Detalles de Implementación y Experimentos

Conclusiones

Conclusiones

Recomendaciones

Recomendaciones

Bibliografía

- Audet, C., Savard, G. & Zghal, W. (2010). A mesh adaptive direct search algorithm for multiobjective optimization. *European Journal of Operational Research*, 204(3), 545-556 (vid. pág. 6).
- Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press. (Vid. pág. 6).
- Custódio, A. L., Madeira, J. A., Vaz, A. I. F. & Vicente, L. N. (2011). Direct multi-search for multiobjective optimization. *SIAM Journal on Optimization*, 21(3), 1109-1140 (vid. pág. 6).
- Deb, K. & Jain, H. (2013). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4), 577-601 (vid. pág. 7).
- Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2), 182-197 (vid. pág. 7).
- Dellnitz, M., Schütze, O. & Hestermeyer, T. (2005). Covering Pareto sets by multi-level subdivision techniques. *Journal of optimization theory and applications*, 124(1), 113-136 (vid. pág. 6).
- Emmerich, M., Beume, N. & Naujoks, B. (2005). An EMO algorithm using the hypervolume measure as selection criterion. *International Conference on Evolutionary Multi-Criterion Optimization*, 62-76 (vid. pág. 7).
- Emmerich, M., Yang, K., Deutz, A., Wang, H. & Fonseca, C. M. (2016). A multi-criteria generalization of bayesian global optimization. *Advances in Stochastic and Deterministic Global Optimization* (pp. 229-242). Springer. (Vid. pág. 6).
- Fleming, P. J., Purshouse, R. C. & Lygoe, R. J. (2005). Many-Objective Optimization: An Engineering Design Perspective. En C. A. Coello Coello, A. Hernández Aguirre & E. Zitzler (Eds.), *Evolutionary Multi-Criterion Optimization* (pp. 14-32). Springer Berlin Heidelberg. (Vid. pág. 5).
- Hillermeier, C. y col. (2001). *Nonlinear multiobjective optimization: a generalized homotopy approach* (Vol. 135). Springer Science & Business Media. (Vid. pág. 6).

- Kuhn, H. W. & Tucker, A. W. (2014). Nonlinear programming. *Traces and emergence of nonlinear programming* (pp. 247-258). Springer. (Vid. pág. 6).
- Mitchell, T., Buchanan, B., DeJong, G., Dietterich, T., Rosenbloom, P. & Waibel, A. (1990). Machine learning. *Annual review of computer science*, 4(1), 417-433 (vid. pág. 1).
- Schütze, O., Dell'Aere, A. & Dellnitz, M. (2005). On Continuation Methods for the Numerical Treatment of Multi-Objective Optimization Problems. En J. Branke, K. Deb, K. Miettinen & R. E. Steuer (Eds.), *Practical Approaches to Multi-Objective Optimization* (pp. 1-15). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. <https://doi.org/10.4230/DagSemProc.04461.16>. (Vid. pág. 6)
Keywords: multi-objective optimization, continuation, k-manifolds
- Zhang, Q. & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6), 712-731 (vid. pág. 7).
- Žilinskas, A. (2013). On the worst-case optimal multi-objective global optimization. *Optimization Letters*, 7(8), 1921-1928 (vid. pág. 6).
- Zitzler, E. & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4), 257-271 (vid. pág. 7).