European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Invited Review

Optimization problems for machine learning: A survey

Claudio Gambella[a,*], Bissan Ghaddar[b], Joe Naoum-Sawaya[b]

[a] IBM Research Ireland, Mulhuddart, Dublin 15, Ireland
[b] Ivey Business School, University of Western Ontario, London, Ontario N6G 0N1, Canada

ARTICLE INFO

Article history:
Received 18 March 2019
Accepted 25 August 2020
Available online xxx

Keywords:
Analytics
Mathematical programming
Machine learning
Deep learning

ABSTRACT

This paper surveys the machine learning literature and presents in an optimization framework several commonly used machine learning approaches. Particularly, mathematical optimization models are presented for regression, classification, clustering, deep learning, and adversarial learning, as well as new emerging applications in machine teaching, empirical model learning, and Bayesian network structure learning. Such models can benefit from the advancement of numerical optimization techniques which have already played a distinctive role in several machine learning settings. The strengths and the shortcomings of these models are discussed and potential research directions and open problems are highlighted.

1. Introduction

The pursuit to create intelligent machines that can match and potentially rival humans in reasoning and making intelligent decisions goes back to at least the early days of the development of digital computing in the late 1950s (Solomonoff, 1957). The goal is to enable machines to perform cognitive functions by learning from past experiences and then solving complex problems under conditions that are varying from past observations. Fueled by the exponential growth in computing power and data collection coupled with the widespread of practical applications, machine learning is nowadays a field of strategic importance.

1.1. Machine learning basics

Broadly speaking, machine learning relies on learning a model that returns the correct output given a certain input. The inputs, i.e., predictor measurements, are typically values that represent the parameters that define a problem, while the output, i.e., response, is a value that represents the solution.

Machine learning models fall into two categories: supervised and unsupervised learning (Friedman, Hastie, & Tibshirani, 2001; James, Witten, Hastie, & Tibshirani, 2013). In supervised learning, a response measurement is available for each observation of predictor measurements and the aim is to fit a model that accurately predicts the response of future observations. More specifically, in supervised learning, values of both the input x and the corresponding output y are available and the objective is to learn a function f that approximates with a reasonable margin of error the relationship between the input and the corresponding output. The accuracy of a prediction is evaluated using a loss function $\mathcal{L}(f(x), y)$ which computes a distance measure between the predicted output and the actual output. In a general setting, the best predictive model f^* is the one that minimizes the risk

$$\mathbb{E}_p[\mathcal{L}(f(x), y)] = \int\int p(x, y)\mathcal{L}(f(x), y)dxdy,$$

where $p(x, y)$ is the probability of observing data point (x, y) (Vapnik, 2013). In practice $p(x, y)$ is unknown, however the assumption is that an independent and identically distributed sample of data points $(x_1, y_1), \ldots, (x_n, y_n)$ forming the training dataset is given. Thus instead of minimizing the risk, the best predictive model f^* is the one that minimizes the empirical risk such that

$$f^* = \arg\min \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(f(x_i), y_i).$$

When learning a model, a key aspect to consider is model complexity. Learning a highly complex model may lead to overfitting, which refers to having a model that fits the training data very well but generalizes poorly to other data. The minimizer of the empirical risk will often lead to overfitting, and hence has a limited generalization property. Furthermore, in practice the data may contain noisy and incorrect values, i.e., outliers, which impacts the value of the empirical risk and subsequently the accuracy of the learned model. Attempting to find a model that perfectly fits every data point in the dataset is thus not desired, since the predictive power

* Corresponding author.
E-mail addresses: claudio.gambella1@ie.ibm.com (C. Gambella), bghaddar@uwaterloo.ca (B. Ghaddar), jnaoumsa@uwaterloo.ca (J. Naoum-Sawaya).

https://doi.org/10.1016/j.ejor.2020.08.045
0377-2217/

of the model will be diminished when points that are far from typical are fitted. Usually, the choice of f is restricted to a family of functions F such that

$$f^* = \arg \min_{f \in F} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x_i), y_i). \quad (1)$$

The degree of model complexity is generally dictated by the nature and size of the training data. While simpler models are advised for small training datasets that do not uniformly cover the possible data ranges, complex models need large data sets to avoid overfitting.

On the other hand, in unsupervised learning, response variables are not available and the goal of learning is to understand the underlying characteristics of the observations. Unsupervised learning thus attempts to learn from the distribution of the data the distinguishing features and the associations in the data. As such, the main use-case for unsupervised learning is exploratory data analysis, where the purpose is to segment and cluster the samples in order to extract insights. While with supervised learning there is a clear measure of accuracy by evaluating the prediction to the known response, in unsupervised it is difficult to evaluate the validity of the derived structure.

The fundamental theory of machine learning models and consequently their success can be largely attributed to research at the interface of computer science, statistics, and operations research. The relation between machine learning and operations research can be viewed along three dimensions: (a) machine learning applied to management science problems, (b) machine learning to solve optimization problems, (c) machine learning problems formulated as optimization problems.

1.2. Machine learning and operations research

Leveraging data in business decision making is nowadays mainstream as any business in today's economy is instrumented for data collection and analysis. While the aim of machine learning is to generate reliable predictions, management science problems deal with optimal decision making. Thus, methodological developments that can leverage data predictions for optimal decision making is an area of research that is critical for future business value (Bertsimas & Kallus, 2020; Kraus, Feuerriegel, & Oztekin, 2020; Mortenson, Doherty, & Robinson, 2015).

Another area of research at the interface of machine learning and operations research is using machine learning to solve hard optimization problems and particularly $\mathcal{NP}$-hard integer constrained optimization (Bonami, Lodi, & Zarpellon, 2018; Khalil, Bodic, Song, Nemhauser, & Dilkina, 2016; Khalil, Dilkina, Nemhauser, Ahmed, & Shao, 2017; Lodi & Zarpellon, 2017; Václavík, Novák, Šucha, & Hanzálek, 2018). In that domain, machine learning models are introduced to complement existing approaches that exploit combinatorial optimization through structure detection, branching, and heuristics.

Lastly, the training of machine learning models can be naturally posed as an optimization problem with typical objectives that include optimizing training error, measure of fit, and cross-entropy (Boţ & Lorenz, 2011; Bottou, Curtis, & Nocedal, 2018; Curtis & Scheinberg, 2017; Wright, 2018). In fact, the widespread adoption of machine learning is in part attributed to the development of efficient solution approaches for these optimization problems, which enabled the training of machine learning models. As we review in this paper, the development of these optimization models has largely been concentrated in areas of computer science, statistics, and operations research. However, diverging publication outlets, standards, and terminology persist.

1.3. Aim and scope

The aim of this paper is to present machine learning as optimization problems. For that, in addition to publications in classical operations research journals, this paper surveys machine learning and artificial intelligence conferences and journals, such as the conference on Association for the Advancement of Artificial Intelligence and the International Conference on Machine Learning. Furthermore, since machine learning research has rapidly accelerated with many important papers still in the review process, this paper also surveys a considerable number of relevant papers that are available on the arXiv repository. This paper also complements the recent surveys of Bottou et al. (2018), Curtis and Scheinberg (2017) and Wright (2018) which described methodological developments for solving machine learning optimization problems; Bengio, Lodi, and Prouvost (2018), and Lodi and Zarpellon (2017) which discussed how machine learning advanced the solution approaches of mathematical programming; Corne, Dhaenens, and Jourdan (2012) and Olafsson, Li, and Wu (2008) which described the interactions between operations research and data mining; Bennett and Parrado-Hernández (2006) which surveyed solution approaches to machine learning models cast as continuous optimization problems; and Song, Triguero, and Özcan (2019) which provided an overview on the various levels of interaction between optimization and machine learning.

This paper presents optimization models for regression, classification, clustering, and deep learning (including adversarial attacks), as well as new emerging paradigms such as machine teaching and empirical model learning. Additionally, this paper highlights the strengths and the shortcomings of the models from a mathematical optimization perspective and discusses potential novel research directions. This is to foster efforts in mathematical programming for machine learning. While important criteria for contributions in operations research are the convergence guarantees, deviation to optimality and speed increments with respect to benchmarks, machine learning applications have a partly different set of goals, such as scalability, reasonable execution time and memory requirement, robustness and numerical stability and, most importantly, generalization (Bennett & Parrado-Hernández, 2006). It is therefore common for mathematical programming approaches to sacrifice optimality (local or global) and convergence guarantees to obtain better generalization properties, by adopting strategies such as early stopping (Poggio et al., 2017).

Following this introductory section, regression models are discussed in Section 2 while classification and clustering models are presented in Sections 3 and 4, respectively. Linear dimension reduction methods are reviewed in Section 5. Deep learning models are presented in Section 6, while models for adversarial learning are discussed in Section 7. New emerging paradigms that include machine teaching and empirical model learning are presented in Section 8. Finally, conclusions are drawn in Section 9.

2. Regression models

2.1. Linear regression

Since the early era of statistics, linear regression models have been widely adopted in supervised learning for predicting a quantitative response. The central assumption is that the relationship between the independent variables (*feature measurements*, or *predictors*, or *input vector*) and the dependent variable (real-valued *output*, or *response*) is representable with a linear function (*regression function*) with a reasonable accuracy. Linear regression models preserve considerable interest, given their simplicity, their extensive range of applications, and the ease of interpretability. In particular, machine learning interpretability, in its simplest form,

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

is the ability to explain in a humanly understandable way the role of the inputs in the outcome (Doshi-Velez & Kim, 2017).

Linear regression aims to find a linear function f that expresses the relation between an input vector x of dimension p and a real-valued output $f(x)$ such as

$$f(x) = \beta_0 + x^\top \beta, \tag{2}$$

where $\beta_0 \in \mathbb{R}$ is the intercept of the regression line and $\beta \in \mathbb{R}^p$ is the vector of coefficients corresponding to each of the input variables. In order to estimate the regression parameters β_0 and β, one needs a training set (X, y) where $X \in \mathbb{R}^{n\times p}$ denotes n training inputs $x_1, \ldots, x_n$ and y denotes n training outputs where each $x_i \in \mathbb{R}^p$ is associated with the real-valued output y_i. The objective is to minimize the empirical risk (1), in order to quantify via β_j the association between predictor X_j and the response, for each $j = 1, \ldots, p$.

The most commonly used loss function for regression is the *least squared estimate*, where fitting a regression model reduces to minimizing the residual sum of squares (RSS) between the labels and the predicted outputs, such as

$$RSS(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2. \tag{3}$$

The least squares estimate is known to have the smallest variance among all linear unbiased estimates, and has a closed form solution. However, this choice is not always ideal for fitting, since it can yield a model with low prediction accuracy, due to a large variance, and often leads to a large number of non-zero regression coefficients (i.e., low interpretability). Shrinkage methods discussed in Section 2.2 and Linear Dimension Reduction discussed in Section 5 are alternatives to the least squared estimate. Forward or backward elimination are also commonly used approaches to perform variable selection and to avoid overfitting (Friedman et al., 2001).

The process of gathering input data is often affected by noise, which can impact the accuracy of statistical learning methods. A model that takes into account the noise in the features of linear regression problems is presented in Bertsimas and Copenhaver (2018), which also investigates the relationship between regularization and robustness to noise. The noise is assumed to vary in an uncertainty set $\mathcal{U} \in \mathbb{R}^{n\times p}$, and the learner adopts the robust perspective:

$$\min_{\beta_0,\beta} \max_{\Delta\in\mathcal{U}} g(y - \beta_0 - (X + \Delta)\beta), \tag{4}$$

where g is a convex function that measures the residuals (e.g., a norm function). The characterization of the uncertainty set $\mathcal{U}$ directly influences the complexity of problem (4).

The design of high-quality linear regression models requires several desirable properties, which are often conflicting and not simultaneously implementable. A fitting procedure based on Mixed Integer Quadratic Programming (MIQP) is presented in Bertsimas and King (2016) and takes into account sparsity, joint inclusion of subset of features (called selective sparsity), robustness to noisy data, stability against outliers, modeler expertise, statistical significance, and low global multicollinearity. Mixed Integer Programming (MIP) models for regression and classification are also investigated in Bertsimas and Shioda (2007). The regression problem is modeled as an assignment of data points to groups with the same regression coefficients.

In order to speed up the fitting procedure and improve the interpretability of the regression model, irrelevant variables can be excluded via feature selection strategies. For example, feature selection is desired in case some regression variables are highly correlated. Multicollinearity can be detected by the condition number of the correlation matrix or the variance influence factor (VIF) (Chatterjee & Hadi, 2015). To achieve feature selection in this case, Tamura et al. (2017) introduces a mixed integer semidefinite programming formulation to eliminate multicollinearity by bounding the condition number. The approach requires to solve a single optimization problem, in contrast with the cutting plane algorithm of Bertsimas and King (2016). Alternatively, Tamura et al. (2019) proposes a mixed integer quadratic optimization formulation with an upper bound on VIF, which is a better-grounded statistical indicator for multicollinearity with respect to the condition number.

2.2. Shrinkage methods

Shrinkage methods (also called regularization methods) seek to diminish the value of the regression coefficients. The aim is to obtain a more interpretable model (with less relevant features), at the price of introducing some bias in the model determination. A well-known shrinkage method is Ridge regression, where a 2-norm penalization on the regression coefficients is added to the loss function such that

$$\mathcal{L}_{ridge}(\beta_0, \beta) = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2, \tag{5}$$

where λ controls the magnitude of shrinkage.

Another technique for regularization in regression is the *lasso regression*, which penalizes the 1-norm of the regression coefficients, and seeks to minimize the quantity

$$\mathcal{L}_{lasso}(\beta_0, \beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|. \tag{6}$$

When λ is sufficiently large, the 1-norm penalty forces some of the coefficient estimates to be exactly equal to zero, hence the models produced by the lasso are more interpretable than those obtained via Ridge regression.

Ridge and lasso regression belong to a class of techniques to achieve *sparse regression*. As discussed in Bertsimas, King, and Mazumder (2016) and Bertsimas, Van Parys et al. (2020), sparse regression can be formulated as the best subset selection problem (Miller, 2002)

$$\min \ \frac{1}{2}\|y - \beta_0 - X\beta\|_2^2 \tag{7}$$

$$\text{s.t } \|\beta\|_0 \le k, \tag{8}$$

$$\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p, \tag{9}$$

where k is an upper bound on the number of predictors with a non-zero regression coefficient, i.e., the predictors to select, and $||\beta||_0$ is the number of non-zero entries of β, which is commonly referred to as the "0-norm" (though is not technically a norm as it does not satisfy the homogeneity property). Problem (7)–(9) is $\mathcal{NP}$-hard, as proven in Natarajan (1995). The recent work of Bertsimas et al. (2016) demonstrated that the best subset selection can be solved to near-optimal solutions using optimization techniques for values of p in the hundreds or thousands. Specifically, by introducing the binary variables $s \in \{0, 1\}^p$, the sparse regression problem can be transformed into the MIQP formulation

$$\min \ \frac{1}{2}\|y - \beta_0 - X\beta\|_2^2 \tag{10}$$

$$\text{s.t } -Ms_j \le \beta_j \le Ms_j \quad \forall j = 1, \ldots, p, \tag{11}$$

$$\sum_{j=1}^{p} s_j \le k, \tag{12}$$

$$\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p, \tag{13}$$

$$s \in \{0, 1\}^p, \tag{14}$$

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

where M is a large constant, $M \geq \|\beta\|_\infty$. Since the choice of the data dependent constant M largely affects the strength of the MIQP formulation, alternative formulations based on Specially Ordered Sets Type I can be devised (Claassen & Hendriks, 2007).

As discussed in Hastie, Tibshirani, and Tibshirani (2017), the prediction accuracy of best subset selection is however highly dependent on the noise present in the input dataset, and it is not possible to establish a dominance relationship over lasso regression and forward stepwise selection (Efron, Hastie, Johnstone, Tibshirani et al., 2004). In order to limit the effect of noise in the input data, make the model more robust, and to avoid numerical issues, Bertsimas et al. (2020) introduces the Tikhonov regularization term $\frac{1}{2\Lambda}\|\beta\|_2^2$ with weight $\Lambda > 0$ into the objective function of problem (10)–(14), which is then solved using a cutting plane approach.

The task of finding a linear model to express the relationship between regressors and output is a particular case of selecting the hyperplane that minimizes a measure of the deviation of the data with respect to the induced linear form. As presented in Blanco, Puerto, and Salmerón (2018), locating a hyperplane $\beta_0 + x^T\beta = 0, \beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p$ to fit a set of points $x_i \in \mathbb{R}^p, i = 1, \ldots, n$, consists of finding $\hat{\beta}_0, \hat{\beta} \in arg\,min_{\beta_0,\beta}\phi(\epsilon(\beta_0, \beta))$, where $\epsilon(\beta_0, \beta) = \epsilon_{\{x_1,\ldots,x_n\}}(\beta_0, \beta)$ is a mapping to the residuals of the points on the hyperplane (according to a distance measure in $\mathbb{R}^p$), and ϕ is an aggregation function on the residuals (e.g., residual sum of squares, least absolute deviation (Edgeworth, 1887)). If the number of points n is much smaller than the dimension p of the space, feature selection strategies can be applied (Bertsimas et al., 2016; Miyashiro & Takano, 2015). We note that hyperplane fitting is a variant of facility location problems (Dıaz-Bánez, Mesa, & Schöbel, 2004; Schöbel, 1998).

2.3. Regression models beyond linearity

A natural extension of linear regression models is to consider nonlinear terms, which may capture complex relationships between regressors and predictors. Nonlinear regression models include, among others, polynomial regression, exponential regression, step functions, regression splines, smoothing splines and local regression (Friedman et al., 2001; James et al., 2013). Alternatively, the Generalized Additive Models (GAMs) (Hastie & Tibshirani, 1986) maintain the additivity of the original predictors $X_1, \ldots, X_p$ and the relationship between each feature and the response y is expressed using nonlinear functions $f_j(X_j)$ such as

$$y = \beta_0 + \sum_{j=1}^{p} f_j(X_j). \tag{15}$$

GAMs may increase the flexibility and accuracy of the predictions with respect to linear models, while maintaining a certain level of interpretability of the predictors. However, one limitation is given by the assumption of additivity of the features. To further increase the model flexibility, one could include predictors of the form $X_i \times X_j$, or consider non-parametric models, such as random forests and boosting. It has been empirically observed that GAMs do not represent well problems where the number of observations is much larger than the number of predictors. In Taylan, Weber, and Beck (2007) the Generalized Additive Model Selection is introduced to fit sparse GAMs in high dimension with a penalized likelihood approach. The penalty term is derived from the fitting criterion for smoothing splines. Alternatively, Chouldechova & Hastie (2015) proposes to fit a constrained version of GAMs by solving a conic programming problem.

As an intermediate model between linear and nonlinear relationships, compact and simple representations via piecewise affine models have been discussed in Keshvari (2018). Piecewise affine forms emerge as candidate models when the fitting function is known to be discontinuous (Ferrari-Trecate, Muselli, Liberati, & Morari, 2003), separable (de Farias, Zhao, & Zhao, 2008), or approximate to complex nonlinear expressions (D'Ambrosio, Lodi, Wiese, & Bragalli, 2015; Rovatti, D'Ambrosio, Lodi, & Martello, 2014; Vielma, Ahmed, & Nemhauser, 2010). Fitting piecewise affine models involves partitioning the domain D of the input data into K subdomains $D_i, i = 1, \ldots, K$, and fitting for each subdomain an affine function $f_i : D_i \to \mathbb{R}$, in order to minimize a measure of the overall fitting error. To facilitate the fitting procedure, the domain is partitioned a priori (see K-hyperplane clustering in Section 4.3). Neglecting domain partitioning may lead to large fitting errors. In contrast, Amaldi, Coniglio, and Taccari (2016) considers both aspects in determining piecewise affine models for piecewise linearly separable subdomains via a mixed integer linear programming formulation and a tailored heuristic. Mixed integer models are also proposed in Toriello and Vielma (2012), however a partial knowledge of the subdomains is required. Alternatively, clustering techniques can be adopted for domain partitioning (Ferrari-Trecate et al., 2003).

3. Classification

The task of classifying data is to decide the class membership of an unlabeled data item x based on the training dataset (X, y) where each x_i has a known class membership y_i. A recent comparison of machine learning techniques for binary classification is found in Baumann, Hochbaum, and Yang (2019). This section reviews the common binary and multiclass classification approaches that include logistic regression, linear discriminant analysis, decision trees, and support vector machines.

3.1. Logistic regression

In most problem domains, there is no functional relationship $y = f(x)$ between y and x. In this case, the relationship between x and y has to be described more generally by a probability distribution $P(x, y)$ while assuming that the training contains independent samples from P. In this section, the label y is assumed to be binary, i.e., $y \in \{0, 1\}$. The optimal class membership decision is to choose the class label y that maximizes the posterior distribution $P(y|x)$. Logistic regression calculates the class membership probability for one of the two categories in the dataset as

$$P(y = 1|x, \beta_0, \beta) = h(x, \beta_0, \beta) = \frac{1}{1 + e^{-(\beta_0 + \beta^\top x)}},$$
$$P(y = 0|x, \beta_0, \beta) = 1 - h(x, \beta_0, \beta).$$

The decision boundary between the two binary classes is formed by a hyperplane whose equation is $\beta_0 + \beta^\top x = 0$. Points at this decision boundary have $P(1|x, \beta_0, \beta) = P(0|x, \beta_0, \beta) = 0.5$. The parameters β_0 and β are usually obtained by maximum-likelihood estimation (Dreiseitl & Ohno-Machado, 2002)

$$\max\ \Pi_{i=1}^{n} P(y_i|x_i, \beta_0, \beta)$$
$$= \max\ \Pi_{i=1}^{n} (h(x_i, \beta_0, \beta))^{y_i} (1 - h(x_i, \beta_0, \beta))^{1-y_i},$$

which is equivalent to

$$\min -\sum_{i=1}^{n} (y_i \log\, h(x_i, \beta_0, \beta) + (1 - y_i)\log(1 - h(x_i, \beta_0, \beta))). \tag{16}$$

Problem (16) is convex and differentiable and first order methods such as gradient descent as well as second order methods such as Newton's method can be applied to find a global optimal solution.

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

To tune the logistic regression model and to avoid overfitting, variable selection can be performed where only the most relevant subsets of the x variables are kept in the model (Friedman et al., 2001). Heuristic approaches such as forward selection or backward elimination can be applied to add or remove variables respectively, based on the statistical significance of each of the computed coefficients. Interaction terms can be also added to further complicate the model at the risk of overfitting the training data.

3.2. Linear discriminant analysis

Linear discriminant analysis (LDA) is an approach for classification and dimensionality reduction. It is often applied to data that contains a large number of features (such as image data) where reducing the number of features is necessary to obtain robust classification. While LDA and Principal Component Analysis (PCA) (see Section 5.1) share the commonality of dimensionality reduction, LDA tends to be more robust than PCA since it takes into account the data labels in computing the optimal projection matrix (Belhumeur, Hespanha, & Kriegman, 1997).

Given the dataset (X, y) where each data sample $x_i \in \mathbb{R}^p$ belongs to one of K classes such that if x_i belongs to the kth class then $y_i(k)$ is 1 where $y_i \in \{0, 1\}^K$, the input data is partitioned into K groups $\{\pi_k\}_{k=1}^K$ where π_k denotes the sample set of the kth class which contains n_k data points. LDA maps the features space $x_i \in \mathbb{R}^p$ to a lower dimensional space $q_i \in \mathbb{R}^r$ $(r < p)$ through a linear transformation $q_i = G^\top x_i$ (Wang, Ding, & Huang, 2010). The class mean of the k-th class is given by $\mu_k = \frac{1}{n_k}\sum_{x_i \in \pi_k} x_i$ while the global mean in given by $\mu = \frac{1}{n}\sum_{i=1}^n x_i$. In the projected space the class mean is given by $\overline{\mu}_k = \frac{1}{n_k}\sum_{q_i \in \pi_k} q_i$ while the global mean in given by $\overline{\mu} = \frac{1}{n}\sum_{i=1}^n q_i$.

The within-class scatter and the between-class scatter evaluate the class separability and are defined as S_w and S_b respectively such that

$$S_w = \sum_{k=1}^{K} \sum_{x_i \in \pi_k} (x_i - \mu_k)(x_i - \mu_k)^\top, \tag{17}$$

$$S_b = \sum_{k=1}^{K} n_k (\mu_k - \mu)(\mu_k - \mu)^\top. \tag{18}$$

The within-class scatter evaluates the spread of the data around the class mean while the between-class scatter evaluates the spread of the class means around the global mean. For the projected data, the within-class and the between-class scatters are defined as $\overline{S}_w$ and $\overline{S}_b$ respectively such that

$$\overline{S}_w = \sum_{k=1}^{K} \sum_{q_i \in \pi_k} (q_i - \overline{\mu}_k)(q_i - \overline{\mu}_k)^\top = G^\top S_w G, \tag{19}$$

$$\overline{S}_b = \sum_{k=1}^{K} n_k (\overline{\mu}_k - \overline{\mu})(\overline{\mu}_k - \overline{\mu})^\top = G^\top S_b G. \tag{20}$$

The LDA optimization problem is bi-objective where the within-class scatter should be minimized while the between-class scatter should be maximized. The optimal transformation G can be obtained by maximizing the Fisher criterion (the ratio of between-class to within-class scatters)

$$\max \frac{|G^T S_b G|}{|G^T S_w G|}. \tag{21}$$

Note that since the between-class and the within-class scatters are not scalar, the determinant is used to obtain a scalar objective function. As discussed in Fukunaga (2013), assuming that S_w is invertible and non-singular, the Fisher criterion is optimized by selecting the r largest eigenvalues of $S_w^{-1}S_b$ and the corresponding eigen vectors $G_1^*, G_2^*, \ldots, G_r^*$ form the optimal transformation matrix $G^* = [G_1^*|G_2^*|\ldots|G_r^*]$. Instead of using Fisher criterion, bi-objective optimization techniques may also potentially be used to formulate and solve the LDA optimization problem exactly.

An alternative formulation of the LDA optimization problem is provided in Chen, Yang, Zhang, and Liang (2013) by maximizing the minimum distance between each class center and the total class center. The proposed approach known as the large margin linear discriminant analysis requires the solution of non-convex optimization problems. A solution approach is also proposed based on solving a series of convex quadratic optimization problems.

3.3. Decision trees

Decision trees are classical models for making a decision or classification using splitting rules organized into a tree data structure. Tree-based methods are non-parametric models that partition the predictor space into sub-regions and then yield a prediction based on statistical indicators (e.g., median and mode) of the segmented training data. Decision trees can be used for both regression and classification problems.

For regression trees, the splitting of the training dataset into distinct and non-overlapping regions can be done using a top-down recursive binary splitting procedure. Starting from a root node that contains the full dataset, a cut that splits the data into distinct sets is identified. For the case of a univariate cut (i.e., involving only a single feature), the cutpoint b for feature j is the one that leads to the two splitted regions $R_1 = \{x_i | x_{ij} < b\}$ and $R_2 = \{x_i | x_{ij} \geq b\}$ that have the greatest possible reduction in the residual sum of squares $\sum_{i:x_i \in R_1(j,b)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,b)} (y_i - \hat{y}_{R_2})^2$, where $\hat{y}_R$ denotes the mean response for the training observations in region R. A multivariate split is of the form $a^T x_i < b$, where a is a vector. Another optimization criterion is the measure of purity (Breiman, Friedman, Olshen, & Stone, 1984) such as Gini's index in classification problems. For classification problems, Breiman et al. (1984) highlights that, given their greedy nature, the classical methods based on recursive splitting do not lead to the global optimality of the decision tree. Since building optimal binary decision trees is known to be $\mathcal{NP}$-hard (Hyafil & Rivest, 1976), heuristic approaches based on mathematical programming paradigms, such as linear optimization (Bennett, 1992), continuous optimization (Bennett & Blue, 1996), and dynamic programming (Azad & Moshkov, 2014a; 2014b; Cox, Qiu, & Kuehner, 1989; Payne & Meisel, 1977), have been proposed.

To find provably optimal decision trees, Bertsimas and Dunn (2017) proposes a mixed integer programming formulation that has an exponential complexity in the depth of the tree. Given a fixed depth D, the maximum number of nodes is $T = 2^{D+1} - 1$ indexed by $t = 1, \ldots, T$. Following the notation of Bertsimas and Dunn (2017), the nodes are split into two sets, branch nodes and leaf nodes. The branch nodes $T_B = \{1, \ldots, \lfloor \frac{T}{2} \rfloor\}$ apply a linear split $a^\top x_i < b$ where the left child node includes the data points that satisfy this split while the right one includes the remaining data. In Bertsimas and Dunn (2017), the splits that are applied at the branch nodes are restricted to a single variable with the option of not splitting a node. The binary decision variable d_t takes a value of 1 if branch node t is split and 0 otherwise. Since the splits are univariate, then variable a_{jt}, which denotes the value of the coefficient of feature j in the split at node t, is also binary. The cutpoint at node t is $b_t \geq 0$.

At each of the leaf nodes $T_L = \{\lfloor \frac{T}{2} \rfloor + 1, \ldots, T\}$, a class prediction is made based on the data points that are included. The binary variable z_{it} indicates if data point i is included to leaf node t, i.e.,

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

$z_{it} = 1$ or otherwise $z_{it} = 0$. The binary decision variable c_{kt} takes a value of 1 if leaf node t is assigned label k, and 0 otherwise while binary variable l_t indicates if leaf node t is used, i.e., $l_t = 1$ or otherwise $l_t = 0$.

The mixed integer programming formulation is

$$\min \ \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} d_t \tag{22}$$

$$\text{s.t.} \quad L_t \geq N_t - N_{kt} - n(1 - c_{kt}), \quad \forall k = 1, \ldots, K, \ t \in T_L, \tag{23}$$

$$0 \leq L_t \leq N_t - N_{kt} + n c_{kt} \quad \forall k = 1, \ldots, K, \ t \in T_L, \tag{24}$$

$$N_{kt} = \frac{1}{2} \sum_{i=1}^{n} (1 + Y_{ik}) z_{it}, \quad \forall k = 1, \ldots, K, \ t \in T_L, \tag{25}$$

$$N_t = \sum_{i=1}^{n} z_{it} \quad \forall t \in T_L, \tag{26}$$

$$\sum_{k=1}^{K} c_{kt} = l_t \quad \forall t \in T_L, \tag{27}$$

$$\sum_{t \in T_L} z_{it} = 1 \quad \forall i = 1, \ldots, n, \tag{28}$$

$$z_{it} \leq l_t \quad \forall i = 1, \ldots, n, \ t \in T_L, \tag{29}$$

$$\sum_{i=1}^{n} z_{it} \geq N_{\min} l_t \quad \forall t \in T_L, \tag{30}$$

$$a_m^\top (x_i + \epsilon) \leq b_m + (1 + \epsilon_{\max})(1 - z_{it}) \quad \forall i = 1, \ldots, n, \ t \in T_L, \ m \in A_L(t), \tag{31}$$

$$a_m^\top x_i \geq b_m - (1 - z_{it}) \quad \forall i = 1, \ldots, n, \ t \in T_L, \ \forall m \in A_R(t), \tag{32}$$

$$\sum_{j=1}^{p} a_{jt} = d_t \quad \forall t \in T_B, \tag{33}$$

$$0 \leq b_t \leq d_t \quad \forall t \in T_B, \tag{34}$$

$$d_t \leq d_{p(t)} \quad \forall t \in T_B \setminus \{1\}, \tag{35}$$

$$z_{it}, \ l_t \in \{0, 1\} \quad \forall i = 1, \ldots, n, \ \forall t \in T_L, \tag{36}$$

$$c_{kt} \in \{0, 1\} \quad \forall k = 1, \ldots, K, \ t \in T_L, \tag{37}$$

$$a_{jt}, \ d_t \in \{0, 1\} \quad \forall j = 1, \ldots, p, \ t \in T_B. \tag{38}$$

The objective function (22) minimizes the normalized total misclassification loss $\frac{1}{\hat{L}} \sum_{t \in T_L} L_t$ and the decision tree complexity which is given by $\sum_{t \in T_B} d_t$, the total number of nodes that are split. α is a tuning parameter and $\hat{L}$ is the baseline loss obtained by predicting the most popular class from the entire dataset. Constraints (23) and (24) set the misclassification loss L_t at leaf node t as $L_t = N_t - N_{kt}$ if node t is assigned label k (i.e $c_{kt} = 1$), where N_t is the total number of data points at leaf node t and N_{kt} is the total number of data points at node t whose true labels are k. The counting of N_{kt} and N_t is enforced by (25) and (26), respectively, where Y_{ik} is a parameter taking the value of 1 if data point i has a label k and -1 otherwise. Constraints (27) indicate that each leaf node that is used (i.e., $l_t = 1$) should be assigned to a label $k = 1 \ldots K$. Constraints (28) indicate that each data point should be assigned to exactly one leaf node. Constraints (29) and (30) indicate that data points can be assigned to a node only if that node is used and if a node is used then at least $N_{\min}$ data points should be assigned to it. The splitting of the data points at each of the branch nodes is enforced by constraints (31) and (32) where $A_L(t)$ is the set of ancestors of t whose left branch has been followed on the path from the root node to node t. Similarly, $A_R(t)$ is the set of ancestors of t whose right branch has been followed on the path from the root node to node t. ϵ and $\epsilon_{\max}$ are small numbers to enforce the strict split $a^\top x < b$ at the left branch (see Bertsimas & Dunn, 2017 for finding good values for ϵ and $\epsilon_{\max}$). Constraints (33) and (34) indicate that the splits are restricted to a single variable with the option of not splitting a node ($d_t = 0$). As enforced by constraints (35), if $p(t)$, the parent of node t, does not apply a split then so is node t. Finally constraints (36)–(38) set the binary conditions.

An alternative formulation to the optimal decision tree problem is provided in Günlük, Kalagnanam, Menickelly, and Scheinberg (2018). The main difference between the formulation of Günlük et al. (2018) and Bertsimas and Dunn (2017) is that the approach of Günlük et al. (2018) is specialized to the case where the features take categorical values. By exploiting the combinatorial structure that is present in the case of categorical variables, (Günlük et al., 2018) provides a strong formulation of the optimal decision tree problem thus improving the computational performance. Furthermore the formulation of (Günlük et al., 2018) is restricted to binary classification and the tree topology is fixed, which lowers the required computational effort for solving the optimization problem to optimality. A commonality between the models presented in Bertsimas and Dunn (2017) and Günlük et al. (2018) is that the split that is considered at each node of the decision tree involves only one variable mainly to achieve better computational performance when solving the optimization model. More generally, splits that span multiple variables can also be considered at each node as presented in Blanquero, Carrizosa, Molero-Rıo, and Morales (2018), Verwer and Zhang (2017) and Verwer, Zhang, and Ye (2017). The approach of Blanquero et al. (2018), which is extended in Blanquero, Carrizosa, Molero-Río, and Morales (2020) to account for sparsity by using regularization, is based on a nonlinear continuous optimization formulation to learn decision trees with general splits.

While single decision tree models are often preferred by data analysts for their high interpretability, the model accuracy can be largely improved by taking multiple decision trees into account. Such approaches include bagging, random forests, and boosting. Bagging creates multiple decision trees by obtaining several training subsets by randomly choosing with replacement data points from the training set and subsequently training a decision tree for each subset. Random forests create training subsets similar to bagging with the addition of randomly selecting a subset of features for training each tree. Boosting iteratively creates decision trees where a weight on the training data is set and is increased at each iteration for the misclassified data points so as to subsequently create a decision tree that is more likely to correctly classify previously misclassified data. These models that make predictions based on aggregating the predictions of individual trees are also known as tree ensemble. A mixed integer optimization model for tree ensemble has been recently proposed in Mišić (2020).

Decision trees can also be used in a more general range of applications as algorithms for problem solving, data mining, and knowledge representation. In Azad and Moshkov (2015), several greedy and dynamic programming approaches are compared for building decision trees on datasets with inconsistent labels (i.e, many-valued decision approach). Many-valued decisions can be evaluated in terms of multiple cost functions in a multi-stage optimization (Azad & Moshkov, 2017). Recently, Chikalov, Hussain, and Moshkov (2018) investigated conflicting objectives in the construction of decision trees by means of bi-criteria optimization. Since the single objectives, such as minimizing average depth or the number of terminal nodes, are known to be $\mathcal{NP}$-hard, the authors

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

propose a bi-criteria optimization approach by means of dynamic programming.

3.4. Support vector machines

Support vector machines (SVMs) are another class of supervised machine learning algorithms that are based on statistical learning and have received significant attention in the optimization literature (Carrizosa & Morales, 2013; Vapnik, 1998; 2013). Given a training set (X, y) with n training inputs where $X \in \mathbb{R}^{n\times p}$ and binary response variables $y \in \{-1, 1\}^n$, the objective of the support vector machine problem is to identify a hyperplane $w^\top x + \gamma = 0$, where $w \in \mathbb{R}^p$ and $\gamma \in \mathbb{R}$, which separates the two classes of data points with a maximal separation margin measured as the width of the band that separates the two classes. In this section, w denotes the vector of coefficients corresponding to each of the input variables and γ is the intercept of the separating hyperplane. As detailed next, the underlying optimization problem is a linearly constrained convex quadratic optimization problem.

3.4.1. Hard margin SVM

The most basic version of SVMs is the hard margin SVM that assumes that there exists a hyperplane that geometrically separates the data points into the two classes such that no data point is misclassified (Cortes & Vapnik, 1995). The training of the SVM model involves finding the hyperplane that separates the data and whose distance to the closest data point in either of the classes, i.e., margin, is maximized.

The distance of a particular data point x_i to the separating hyperplane is

$$\frac{y_i(w^\top x_i + \gamma)}{\|w\|_2}.$$

The distance to the closest data point is normalized to $\frac{1}{\|w\|_2}$ where $\|w\|_2$ denotes the 2-norm. Thus the data points with labels $y = -1$ are on one side of the hyperplane such that $w^\top x + \gamma \le 1$ while the data point with labels $y = 1$ are on the other side $w^\top x + \gamma \ge 1$. The optimization problem for finding the separating hyperplane is then

$$\begin{aligned} \max\ & \frac{1}{\|w\|_2} \\ \text{s.t. } & y_i(w^\top x_i + \gamma) \ge 1 \quad \forall i = 1, \dots, n, \\ & w \in \mathbb{R}^p, \gamma \in \mathbb{R}, \end{aligned}$$

which is equivalent to

$$\min \ \|w\|_2^2 \tag{39}$$

$$\text{s.t. } y_i(w^\top x_i + \gamma) \ge 1 \quad \forall i = 1, \dots, n, \tag{40}$$

$$w \in \mathbb{R}^p, \gamma \in \mathbb{R}, \tag{41}$$

that is a convex quadratic problem.

Forcing the data to be separable by a linear hyperplane is a strong condition that often does not hold in practice. Thus, the soft-margin SVM, which relaxes the condition of perfect separability, is used instead.

3.4.2. Soft-Margin SVM

When the data is not linearly separable, problem (39)–(41) is infeasible. Alternatively, Bennett and Mangasarian (1992) proposed a linear program that minimizes a weighted average of the errors given by the points lying on the wrong side of the separator. This work was then extended in Cortes and Vapnik (1995) which presented the soft margin SVM. The soft margin SVM introduces slack variables $\xi_i \ge 0$ into constraints (40) which are then penalized in the objective function as a proxy to minimizing the number of data points that are on the wrong side. The soft-margin SVM optimization problem is

$$\min \ \|w\|_2^2 + C\sum_{i=1}^{n} \xi_i \tag{42}$$

$$\text{s.t. } y_i(w^\top x_i + \gamma) \ge 1 - \xi_i \quad \forall i = 1, \dots, n, \tag{43}$$

$$w \in \mathbb{R}^p, \gamma \in \mathbb{R}, \tag{44}$$

$$\xi_i \ge 0 \quad \forall i = 1, \dots, n. \tag{45}$$

Another common alternative is to include the error term ξ_i in the objective function by using the squared hinge loss $\sum_i^n \xi_i^2$ instead of the hinge loss $\sum_i^n \xi_i$. The hinge loss function takes a value of zero for a data point that is correctly classified while it takes a positive value that is proportional to the distance from the separating hyperplane for a misclassified data point. Hyperparameter C is then tuned to obtain the best classifier.

Besides the direct solution of problem (42)–(45) as a convex quadratic problem, replacing the 2-norm by the 1-norm leads to a linear optimization problem generally at the expense of higher misclassification rate (Bradley & Mangasarian, 2000).

3.4.3. Sparse SVM

Using the 1-norm is also an approach to sparsify w, i.e., reduce the number of features that are involved in the classification model (Bradley & Mangasarian, 2000; Zhu, Rosset, Tibshirani, & Hastie, 2004). An approach known as the elastic net includes both the 1-norm and the 2-norm in the objective function and tunes the bias towards one of the norms through a hyperparameter (Wang, Zhu, & Zou, 2006; Zou & Hastie, 2005). Several other approaches for dealing with sparsity in SVM have been proposed in Aytug (2015), Dunbar, Murray, Cysique, Brew, and Jeyakumar (2010), Gaudioso, Gorgone, Labbé, and Rodríguez-Chía (2017), Ghaddar and Naoum-Sawaya (2018), Guyon, Weston, Barnhill, and Vapnik (2002), Maldonado, Pérez, Weber, and Labbé (2014), and Piramuthu (2004). The number of features can be explicitly modeled in (42)–(45) by using binary variables $z \in \{0, 1\}^p$ where $z_j = 1$ indicates that feature j is selected and otherwise $z_j = 0$ (Chan, Vasconcelos, & Lanckriet, 2007). A constraint limiting the number of features to a maximum desired number can be enforced resulting in the following mixed integer quadratic problem

$$\min \ \|w\|_2^2 + C\sum_{i=1}^{n} \xi_i \tag{46}$$

$$\text{s.t. } y_i(w^\top x_i + \gamma) \ge 1 - \xi_i \quad \forall i = 1, \dots, n, \tag{47}$$

$$-Mz_j \le w_j \le Mz_j \quad \forall j = 1, \dots, p, \tag{48}$$

$$\sum_{j=1}^{p} z_j \le r, \tag{49}$$

$$w \in \mathbb{R}^p, \gamma \in \mathbb{R}, \tag{50}$$

$$z_j \in \{0, 1\} \quad \forall j = 1, \dots, p, \tag{51}$$

$$\xi_i \ge 0 \quad \forall i = 1, \dots, n. \tag{52}$$

Constraints (48) force $z_j = 1$ when feature j is used, i.e., $w_j \ne 0$ (M denotes a sufficiently large number). Constraints (49) set a limit r on the maximum number of features that can be used.

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

3.4.4. The dual problem and kernel tricks

The data points can be mapped to a higher dimensional space through a mapping function $\phi(x)$. A soft margin SVM can then be applied such that

$$\min \ \|w\|_2^2 + C\sum_{i=1}^{n}\xi_i \tag{53}$$

$$\text{s.t.} \quad y_i(w^\top\phi(x_i)+\gamma) \geq 1-\xi_i \quad \forall i=1,\dots,n, \tag{54}$$

$$w\in\mathbb{R}^p, \gamma\in\mathbb{R}, \tag{55}$$

$$\xi_i \geq 0 \quad \forall i=1,\dots,n. \tag{56}$$

Through this mapping, the data has a linear classifier in the higher dimensional space however a nonlinear separation function is obtained in the original space.

To solve problem (53)–(56), the following dual problem is first obtained

$$\max_{\alpha} \ \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \phi(x_i)^\top\phi(x_j)$$

$$\text{s.t.} \ \sum_{i=1}^{n}\alpha_i y_i = 0 \quad \forall i=1,\dots,n,$$

$$0\leq\alpha_i\leq C \quad \forall i=1,\dots,n,$$

where α_i are the dual variables of constraints (54). Given a kernel function $K:\mathbb{R}^m\times\mathbb{R}^m\to\mathbb{R}$ where $K(x_i,x_j)=\phi(x_i)^\top\phi(x_j)$, the dual problem is

$$\max_{\alpha} \ \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_i,x_j)$$

$$\text{s.t.} \ \sum_{i=1}^{n}\alpha_i y_i = 0, \quad \forall i=1,\dots,n,$$

$$0\leq\alpha_i\leq C, \quad \forall i=1,\dots,n,$$

which is a convex quadratic optimization problem. Thus only the kernel function $K(x_i, x_j)$ is required while the explicit mapping ϕ is not needed.

Among the commonly used kernel functions is the polynomial $K(x_i,x_j)=(x_i^\top x_j+c)^d$ where c controls the trade-off between the higher-order and the lower-order terms in the polynomial and d is the degree of the polynomial. The polynomial kernel models the interaction between the data up to the degree d. A high degree polynomial tends to overfit the training data. Another kernel function is the radial basis $K(x_i,x_j)=e^{-\frac{\|x_i-x_j\|_2^2}{\gamma}}$ where γ acts as a smoothing parameter. A smaller γ tends to overfit the training data. The sigmoidal kernel $K(x_i,x_j)=\tanh(\varphi x_i^\top x_j+c)$ is also commonly used where φ is a scaling parameter of the input data and c is a shifting parameter that controls the threshold of the mapping. Further details on kernel functions are provided in Alam, Lin, Deng, Calhoun, and Wang (2018), Carrizosa and Morales (2013), and Herbrich (2001).

Since the classification in high dimensional space can be difficult to interpret for practitioners, Binarized SVM (BSVM) replaces the continuous predictor variables with a linear combination of binary cutoff variables Carrizosa, Martín-Barragán, and Morales (2010). BSVM is also extended in Carrizosa, Martín-Barragán, and Morales (2011) to capture the interactions between the relevant variables. Another important practical aspect to consider is data uncertainty. Often the training data suffers from inaccuracies in the labels and in the features that are collected which may negatively affect the performance of the classifiers. While typically regularization is used to mitigate the effect of uncertainty, Bertsimas, Dunn, Pawlowski, and Zhuo (2019) proposes robust optimization models for logistic regression, decision trees, and support vector machines and shows increased accuracy over regularization, and most importantly without changing the complexity of the classification problem.

3.4.5. Support vector regression

Although as discussed earlier, SVM has been introduced for binary classification, its extension to regression, i.e., support vector regression, has received significant interest in the literature (Smola & Schölkopf, 2004). The core idea of support vector regression is to find a linear function $f(x)=w^\top x+\gamma$ that can approximate with a tolerance ϵ a training set (X, y) where $y\in\mathbb{R}$ (Vapnik, 2013). Such a linear function may however not exist, and thus slack variables $\xi_i^+\geq 0$ and $\xi_i^-\geq 0$ denoting positive and negative deviations from the desired tolerance are introduced and minimized similar to the soft-margin SVM. The corresponding optimization problem is

$$\min \ \|w\|_2^2 + C\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-) \tag{57}$$

$$\text{s.t.} \quad y_i - w^\top x_i - \gamma \leq \epsilon + \xi^+ \quad \forall i=1,\dots,n, \tag{58}$$

$$w^\top x_i + \gamma - y_i \leq \epsilon + \xi^- \quad \forall i=1,\dots,n, \tag{59}$$

$$w\in\mathbb{R}^p, \gamma\in\mathbb{R}, \tag{60}$$

$$\xi_i^+, \ \xi_i^- \geq 0 \quad \forall i=1,\dots,n. \tag{61}$$

Hyperparameter C is tuned to adjust the weight on the deviation from the tolerance ϵ. This deviation from ϵ is the ϵ-insensitive loss function $|\xi|_\epsilon$ given by

$$|\xi|_\epsilon = \begin{cases} 0 & \text{if } |\xi|\leq\epsilon, \\ |\xi|-\epsilon & \text{otherwise.} \end{cases}$$

As detailed extensively in Smola and Schölkopf (2004), kernel tricks can also be applied to (57)–(61) which is solved by formulating the dual problem.

3.4.6. Support vector ordinal regression

In situations where the data contains ordering preferences, i.e., the training data is labeled by ranks, where the order of the rankings is relevant while the distances between the ranks is not defined or irrelevant to the training, the purpose of learning is to find a model that maps the preference information.

The application of classic regression models for such type of data requires the transformation of the ordinal ranks to numerical values. However, such approaches often fail in providing robust models as an appropriate function to map the ranks to distances is challenging to find (Kramer, Widmer, Pfahringer, & De Groeve, 2001). An alternative is to encode the ordinal ranks into binary classifications at the expense of a large increase in the scale of the problems (Har-Peled, Roth, & Zimak, 2003; Herbrich, Graepel, & Obermayer, 2000).

An extension of SVM for ordinary data has been proposed in Shashua and Levin (2003) and extended in Chu and Keerthi (2007). Given a training dataset with r ordered categories $\{1,\dots,r\}$ where n_j is the number of data points labeled as order j, the support vector ordinal regression finds $r-1$ separating parallel hyperplanes $w^\top x+\beta_j=0$ where β_j is the threshold associated with the hyperplane that separates the orders $k\leq j$ from the remaining orders. Thus $x_{i,k}$, the i^{th} data sample of order $k\leq j$, should have a function value lower than the margin β_j-1 while the data samples with orders $k>j$ should have a function value greater than the margin β_j+1. The errors for violating these conditions are given

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

by $\xi^+_{i,kj} \geq 0$ and $\xi^-_{i,kj} \geq 0$ respectively. Following Chu and Keerthi (2007), the associated SVM formulation is

$$\begin{aligned}
\min \quad & \|w\|_2^2 + C \sum_{j=1}^{r-1} \left(\sum_{k=1}^{j} \sum_{i=1}^{n_k} \xi^+_{i,kj} + \sum_{k=j+1}^{r} \sum_{i=1}^{n_k} \xi^-_{i,kj} \right) \\
\text{s.t.} \quad & w^\top x_{i,k} - \beta_j \leq -1 + \xi^+_{i,kj} \\
& \quad \forall k = 1, \ldots, j, \ \ j = 1, \ldots, r-1, \ i = 1, \ldots, n_k, \\
& w^\top x_{i,k} - \beta_j \geq 1 - \xi^-_{i,kj} \\
& \quad \forall k = j+1, \ldots, r, \ \ j = 1, \ldots, r-1, \ \ i = 1, \ldots, n_k, \\
& w \in \mathbb{R}^p, \ \ \beta_j \in \mathbb{R} \quad \forall j = 1, \ldots, r-1, \\
& \xi^+_{i,kj} \geq 0 \quad \forall k = 1, \ldots, j, \ \ j = 1, \ldots, r-1, \ \ i = 1, \ldots, n_k, \\
& \xi^-_{i,kj} \geq 0 \quad \forall k = j+1, \ldots, r, \ \ j = 1, \ldots, r-1, \ \ i = 1, \ldots, n_k.
\end{aligned}$$

As detailed in Chu and Keerthi (2007), kernel tricks can be also applied by considering the dual problem. Finally we note that preference modeling using machine learning has several commonalities with various approaches in multi-criteria decision analysis and most notably, robust ordinal regression. We refer the readers to Corrente, Greco, Kadziński, and Słowiński (2013) for a detailed comparison between preference learning using machine learning and muti-criteria decision making.

4. Clustering

Data clustering is a class of unsupervised learning approaches that has been widely used, particularly in applications of data mining, pattern recognition, and information retrieval. Given an input $X \in \mathbb{R}^{n \times p}$, which includes n unlabeled observations $x_1, \ldots, x_n$ with $x_i \in \mathbb{R}^p$, cluster analysis aims at finding K subsets of X, called clusters, which are homogeneous and well separated. *Homogeneity* indicates the similarity of the observations within the same cluster (typically, by means of a distance metric), while the *separability* accounts for the differences between entities of different clusters. The two concepts can be measured via several criteria and lead to different types of clustering algorithms (see, e.g., Hansen & Jaumard, 1997). The number of clusters is typically a tuning parameter to be fixed before determining the clusters. An extensive survey on data clustering analysis is provided in Jain, Murty, and Flynn (1999).

In case the entities are points in a Euclidean space, the clustering problem is often modeled as a network problem and shares many similarities with classical problems in operations research, such as the p-median problem (Benati & García, 2014; Klastorin, 1985; Mai, Fry, & Ohlmann, 2018; Mulvey & Crowder, 1979). In the following subsections, the commonly used minimum sum-of-squares clustering, the capacitated clustering, and the K-hyperplane clustering are discussed.

4.1. Minimum sum-of-squares clustering (a.k.a. K-means clustering)

Minimum sum-of-squares clustering is one of the most commonly adopted clustering algorithms. It requires to find a number of disjoint clusters for observations $x_i, i = 1, \ldots, n$, where $x_i \in \mathbb{R}^p$ such that the distance to cluster centroids is minimized. Given that typically the number of clusters K is a-priori fixed, the problem is also referred to as K-means clustering. The decision of the cluster size is typically taken by examining the elbow curve, or similarity indicators, such as silhouette values and Calinski-Harabasz index, or via mathematical programming approaches including the maximization of the modularity of the associated graph (Cafieri, Costa, & Hansen, 2014a; Cafieri, Hansen, & Liberti, 2014b).

Defining the binary variables

$$u_{ij} = \begin{cases} 1 & \text{if observation i belongs to cluster j} \\ 0 & \text{otherwise,} \end{cases}$$

and the centroid $\mu_j \in \mathbb{R}^p$ of each cluster j, the problem of minimizing the within-cluster variance is formulated in Aloise, Hansen, and Liberti (2012) as the following mixed integer nonlinear program

$$\min \sum_{i=1}^{n} \sum_{j=1}^{K} u_{ij} \|x_i - \mu_j\|_2^2 \tag{62}$$

$$\text{s.t.} \quad \sum_{j=1}^{K} u_{ij} = 1 \quad \forall i = 1, \ldots, n, \tag{63}$$

$$\mu_j \in \mathbb{R}^p \quad \forall j = 1, \ldots, K, \tag{64}$$

$$u_{ij} \in \{0, 1\} \quad \forall i = 1, \ldots, n, \ \ j = 1, \ldots, K. \tag{65}$$

By introducing the variables d_{ij} which denote the distance of observation i from centroid j, the following linearized formulation is obtained

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} \sum_{j=1}^{K} d_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{K} u_{ij} = 1, \quad \forall i = 1, \ldots, n, \\
& d_{ij} \geq ||x_i - \mu_j||_2^2 - M(1 - u_{ij}) \quad \forall i = 1, \ldots, n, \ \ j = 1, \ldots, K, \\
& \mu_j \in \mathbb{R}^p \quad \forall j = 1, \ldots, K, \\
& u_{ij} \in \{0, 1\}, \ \ d_{ij} \geq 0 \quad \forall i = 1, \ldots, n, \ \ j = 1, \ldots, K.
\end{aligned}$$

Parameter M is a sufficiently large number. A heuristic solution approach based on the gradient method is proposed for problem (62)–(65) in Bagirov and Yearwood (2006). Alternatively, a column generation approach for large-scale instances has been proposed in Aloise et al. (2012) and a bundle approach has been presented in Karmitsa, Bagirov, and Taheri (2017).

The case where the space is not Euclidean is considered in Carrizosa, Mladenović, and Todosijević (2013). Alternatively, Santi, Aloise, & Blanchard (2016) presents the Heterogeneous Clustering Problem (HCP) where the observations to cluster are associated with multiple dissimilarity matrices. HCP is formulated as a mixed integer quadratically constrained quadratic program. Another variant is presented in Sağlam, Salman, Sayın, and Türkay (2006) where the homogeneity is expressed by the minimization of the maximum diameter $D_{\max}$ of the clusters. The resulting nonconvex bilinear mixed integer program is solved via a graph-theoretic approach based on seed finding.

Many common solution approaches for K-means clustering are based on heuristics. A popular method implemented in data science packages (e.g., scikit-learn Pedregosa et al., 2011) is the two-step improvement procedure proposed in MacQueen (1967). Starting from a sample of K points in set X as initial cluster centers (centroids μ_j^0), at each iteration k, the algorithm assigns each point in X to the nearest centroid μ_j^k and then computes the centroids μ_j^{k+1} of the new partition. The procedure is guaranteed to decrease the within-cluster variance and it is run until this metric is sufficiently low. Given the dependency of the procedure to the choice of μ_j^0, typically the clustering is repeated with different initial centroids and the best clusters are selected. Other heuristics relax the assumption to produce exactly K clusters. For instance, MacQueen (1967) merges clusters if their centroids are sufficiently close. Clustering is also used within heuristics for hard combinatorial problems (Ganesh & Narendran, 2007; Kwatera & Simeone, 1993), and

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

can be integrated in problems where the evaluation of multiple solutions is important (e.g., Cluster Newton Method Aoki, Hayami, Sterck, & Konagaya, 2014; Gaudreau, Hayami, Aoki, Safouhi, & Konagaya, 2015). Cluster Newton method approximates the Jacobian in the domain covered by the cluster of points, instead of locally as done by the traditional Newton's Method (Kelley, 1999), and this has a regularization effect.

4.2. Capacitated clustering

The Capacitated Centered Clustering Problem (CCCP) deals with finding a set of clusters with a capacity limitation and homogeneity expressed by the similarity to the cluster centre. Given a set of potential clusters $1, \ldots, K$, a mathematical formulation for CCCP is given in Negreiros and Palhano (2006) as

$$\min \sum_{i=1}^{n}\sum_{j=1}^{K} \underline{s}_{ij} u_{ij} \tag{66}$$

$$\text{s.t} \quad \sum_{j=1}^{K} u_{ij} = 1 \quad \forall i = 1, \ldots, n, \tag{67}$$

$$\sum_{j=1}^{K} v_j \leq K, \tag{68}$$

$$u_{ij} \leq v_j \quad \forall i = 1, \ldots, n, \;\; j = 1, \ldots, V, \tag{69}$$

$$\sum_{i=1}^{n} q_i u_{ij} \leq Q_j \quad \forall j = 1, \ldots, K, \tag{70}$$

$$u_{ij}, v_j \in \{0, 1\} \quad \forall i = 1, \ldots, n, \;\; j = 1, \ldots, K.$$

Parameter K is an upper bound on the number of clusters, $\underline{s}_{ij}$ is the dissimilarity measure between observation i and cluster j, q_i is the weight of observation i, and Q_j is the capacity of cluster j. Variable u_{ij} denotes the assignment of observation i to cluster j and variable v_j is equal to 1 if cluster j is used. If the metric $\underline{s}_{ij}$ is a distance and the clusters are homogeneous (i.e., $Q_j = Q \; \forall j$), the formulation also models the well-known facility location problem. A solution approach is discussed in Chaves and Lorena (2010) while an alternative quadratic programming formulation is presented in Lewis, Wang, and Kochenberger (2014). Solution heuristics have also been proposed in Mai et al. (2018) and Scheuerer and Wendolsky (2006).

4.3. K-*hyperplane clustering*

In the K-Hyperplane Clustering (K-HC) problem, a hyperplane, instead of a center, is associated with each cluster. This is motivated by applications such as text mining and image segmentation, where collinearity and coplanarity relations among the observations are the main interest of the unsupervised learning task, rather than the similarity. Given the observations $x_i, i = 1, \ldots, n$, the K-HC problem requires to find K clusters, and a hyperplane $H_j = \{x \in \mathbb{R}^p : w_j^T x = \gamma_j\}$, with $w_j \in \mathbb{R}^p$ and $\gamma_j \in \mathbb{R}$, for each cluster j. The aim is to minimize the sum of the squared 2-norm Euclidean orthogonal distances between each observation and the corresponding cluster.

Given that the orthogonal distance of x_i to hyperplane H_j is given by $\frac{|w_j^T x_i - \gamma_j|}{\|w\|_2}$, K-HC is formulated in Amaldi and Coniglio (2013) as the following mixed integer quadratically constrained quadratic problem

$$\min \sum_{i=1}^{n} \delta_i^2 \tag{71}$$

$$\text{s.t} \quad \sum_{j=1}^{K} u_{ij} = 1 \quad \forall i = 1, \ldots, n, \tag{72}$$

$$\delta_i \geq (w_j^T x_i - \gamma_j) - M(1 - u_{ij}) \quad \forall i = 1, \ldots, n, \;\; j = 1, \ldots, K, \tag{73}$$

$$\delta_i \geq (-w_j^T x_i + \gamma_j) - M(1 - u_{ij}) \quad \forall i = 1, \ldots, n, \;\; j = 1, \ldots, K, \tag{74}$$

$$\|w_j\|_2 \geq 1 \quad \forall j = 1, \ldots, K, \tag{75}$$

$$\delta_i \geq 0 \quad \forall i = 1, \ldots, n, \tag{76}$$

$$w_j \in \mathbb{R}^p, \gamma_j \in \mathbb{R} \quad \forall j = 1, \ldots, K, \tag{77}$$

$$u_{ij} \in \{0, 1\} \quad \forall i \in 1, \ldots, n, \;\; j = 1, \ldots, K. \tag{78}$$

Binary variable u_{ij} is equal to 1 if point x_i is assigned to cluster j, and 0 otherwise. Linear constraints (73)–(74) set δ_i as the distance between point x_i and the hyperplane of cluster j. These constraints are enforced only if u_{ij} is equal to 1, otherwise they are redundant. The non-convexity is due to constraint (75). As a solution approach, a distance-based reassignment heuristic that outperforms spatial branch-and-bound solvers is proposed in Amaldi and Coniglio (2013).

5. Linear dimension reduction

In Section 2.2, shrinkage methods have been discussed as a way to improve model interpretability by fitting a model with all original p predictors. In this section, we discuss dimension reduction methods that search for $H < p$ linear combinations of the predictors such that $Z_h = \sum_{j=1}^{p} \phi_j^h X_j$ (also called *projections*) where X_j denotes column j of X, i.e., the vector of values of feature j of the training set. While this section focuses on Principal Component Analysis and Partial Least Squares, we note that other linear and nonlinear dimension reduction methods exist. An extensive survey on benefits and shortcomings of dimension reduction methods is presented in Cunningham and Ghahramani (2015).

5.1. Principal components

Principal Components Analysis (PCA) (Jolliffe, 2011) aims to find a low-dimensional representation of the dataset with highly informative derived features. Principal components are ordered in terms of their explained variances, which measure the amount of information retained from the original set of features $X_1, \ldots, X_p$.

In particular, assuming the regressors are standardized to a mean of 0 and a variance of 1, the direction of the first principal component is a unit vector $\phi^1 \in \mathbb{R}^p$ that is the solution of the optimization problem

$$\max_{\phi^1 \in \mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p} \phi_j^1 x_{ij}\right)^2 \tag{79}$$

$$\text{s.t.} \sum_{j=1}^{p} (\phi_j^1)^2 = 1. \tag{80}$$

Problem (79)–(80) is the traditional formulation of PCA and can be solved via Lagrange multipliers methods. Since the formulation is sensitive to the presence of outliers, several approaches have been proposed to improve robustness (Reris & Brooks, 2015). One approach is to replace the 2-norm in (79) with the 1-norm.

An iterative approach can be used to obtain the principal components where the first principal component $Z_1 = \sum_{j=1}^{p} \phi_j^1 X_j$ is the projection of the original features with the largest variability. The subsequent principal components are obtained iteratively where each principal component $Z_h, h = 2, \ldots, H$ is obtained by a linear

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

combination of the feature columns $X_1, \ldots, X_p$. Each Z_h is uncorrelated with $Z_1, \ldots, Z_{h-1}$ which have larger variance. Introducing the sample covariance matrix S of the regressors X_j, the direction $\phi^h \in \mathbb{R}^p$ of the h-th principal component Z_h is the solution of

$$\max_{\phi^h \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_j^h x_{ij} \right)^2 \tag{81}$$

$$\text{s.t.} \quad \sum_{j=1}^{p} (\phi_j^h)^2 = 1, \tag{82}$$

$$\phi^{h\top} S \phi^l = 0 \quad \forall l = 1, \ldots, h-1. \tag{83}$$

PCA can be used for several data analysis problems which benefit from reducing the problem dimension. Principal Components Regression (PCR) is a two-stage procedure that uses the first principal components as predictors for a linear regression model. PCR has the advantage of including less predictors than the original set and at the same time retaining the variability of the dataset in the derived features. However, principal components might not be relevant with the response variables of the regression.

To select principal components in regression models, the regression loss function and the PCA objective function can be combined in a single-step quadratic programming formulation (Kawano, Fujisawa, Takada, & Shiroishi, 2015).

Since the identification of the principal components does not require any knowledge of the response y, PCA can also be adopted in unsupervised learning such as in the k-means clustering method (see Section 4.1, (Ding & He, 2004)). A known drawback of PCA is interpretability. To promote the sparsity of the projected components, and thus make them more interpretable, Carrizosa and Guerrero (2014b) formulates a Mixed Integer Nonlinear Programming (MINLP) problem and shows that the level of sparsity can be imposed in the model. Alternatively, the variance of the principal components and their sparsity can be jointly maximized in a biobjective framework (Carrizosa & Guerrero, 2014a).

5.2. Partial least squares

Partial Least Squares (PLS) identifies transformed features $Z_1, \ldots, Z_H$ by projecting both the predictors X and their corresponding response y into a new space, and this is an approach specific to regression problems (Friedman et al., 2001). PLS is particularly viable for problems with a large number of features compared to observations as it aims to identify the latent factors that explain most the variations in the response. PLS corresponds to fitting simple regression models each containing a single predictor variable.

The first PLS direction is denoted by $\phi^1 \in \mathbb{R}^p$ where each component ϕ_j^1 is found by fitting a regression with predictor X_j and response y. The first PLS direction points towards the features that are more strongly related to the response. For computing the second PLS direction, the features vectors $X_1, \ldots, X_p$ are first orthogonalized with respect to Z_1 (as per the Gram-Schmidt approach), and then individually fitted in simple regression models with response y. The process is iterated for all PLS directions $H < p$. The coefficient of the simple regression of y onto each original feature X_j can also be computed as the inner product $\langle y, X_j \rangle$. Similar to PCR, PLS then fits a linear regression model with regressors $Z_1, \ldots, Z_H$ and response y.

While the principal components directions maximize variance, PLS searches for directions $Z_h = \sum_{j=1}^{p} \phi_j^h X_j$ with both high variance and high correlation with the response. The hth direction ϕ^h can be found by solving the optimization problem

$$\max_{\phi^h \in \mathbb{R}^p} \text{Corr}(y, X\phi^h)^2 \times \text{Var}(X\phi^h) \tag{84}$$

$$\text{s.t.} \quad \sum_{j=1}^{p} (\phi_j^h)^2 = 1, \tag{85}$$

$$\phi^{h\top} S \phi^l = 0 \quad \forall l = 1, \ldots, h-1, \tag{86}$$

where Corr() indicates the correlation matrix, Var() the variance, S the sample covariance matrix of X_j, and (86) ensures that Z_m is uncorrelated with the previous directions $Z_l = \sum_{j=1}^{p} \phi_j^l X_j$.

Table 1
Notation for DNN architectures.

$\{0, \ldots, L\}$	*layers* indices.
n^l	number of *units*, or *neurons*, in layer l.
σ	element-wise activation function.
$U(j, l)$	jth unit of layer l.
$W^l \in \mathbb{R}^{n^l \times n^{l+1}}$	weight matrix for layer $l < L$.
$b^l \in \mathbb{R}^{n^l}$	bias vector for layer $l > 0$.
(X, y)	*training* dataset, with observations x_i and responses y_i, $i = 1, \ldots, n$.
x^l	output vector of layer l ($l = 0$ indicates *input feature* vector, $l > 0$ indicates *derived feature* vector).

6. Deep learning

Deep Learning received a first momentum until the 80s due to universal approximation results (Cybenko, 1989; Hornik, 1991). Neural networks with a single layer with a finite number of units can represent any multivariate continuous function on a compact subset in $\mathbb{R}^n$ with arbitrary precision. However, the computational complexity required for training Deep Neural Networks (DNNs) hindered their diffusion by late 90s. Starting 2010, the empirical success of DNNs has been widely recognized for several reasons, including the development of advanced processing units, namely GPUs, the advances in the efficiency of training algorithms such as backpropagation, the establishment of proper initialization parameters, and the massive collection of data enabled by new technologies in a variety of domains (e.g., healthcare, supply chain management (Tiwari, Wee, & Daryanto, 2018), marketing, logistics (Wang, Gunasekaran, Ngai, & Papadopoulos, 2016), Internet of Things).

DNNs can be used for the regression and classification tasks discussed in the previous sections, especially when traditional machine learning models fail to capture complex relationships between the input data and the quantitative response, or class, to be learned. The aim of this section is to describe the decision optimization problems associated with DNN architectures. To facilitate the presentation, the notation for the common parameters is provided in Table 1, and an example of fully connected feedforward network is shown in Fig. 1.

The output vector x^L of a DNN is computed by propagating the information from the input layer to each following layer via the weight matrices $W^l, l < L$, the bias vectors $b^l, l > 0$, and the activation function σ, such that

$$x^l = \sigma(W^{l-1} x^{l-1} + b^{l-1}) \quad \forall l = 1, \ldots, L. \tag{87}$$

Activation functions indicate whether a neuron should be activated or not in the network, and are responsible for the capability of DNNs to learn complex relationships between the input and the output. The *rectified linear unit*

$$ReLU : \mathbb{R}^n \to \mathbb{R}^n, ReLU(z) = (\max(0, z_1), \ldots, \max(0, z_n))$$

is typically one of the preferred options for activation functions, mainly because it can be optimized with gradient-based methods for DNN training, and tends to produce sparse networks (where not all neurons are activated).

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

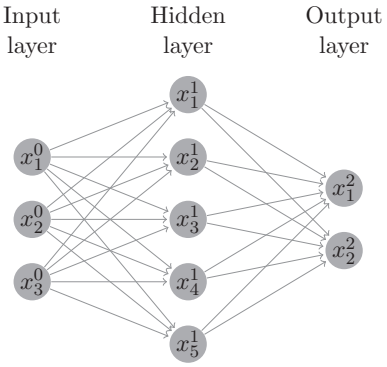

Fig. 1. Deep Feedforward Neural Network with 3 layers. The input layer has $n^0 = 3$ units, the hidden layer has $n^1 = 5$ units and there are $n^2 = 2$ output units. This is an example of fully connected network, where each neuron in one layer is connected to all neurons in the next layer. Training such network requires to determine weight matrices $W^0 \in \mathbb{R}^{3\times5}$, $W^1 \in \mathbb{R}^{5\times2}$, and bias vectors $b^1 \in \mathbb{R}^5$, $b^2 \in \mathbb{R}^2$.

In the context of regression, the components of x^L can directly represent the response values learned. For a classification problem, the vector x^L corresponds to the *logits* of the classifier. In order to interpret x^L as a vector of class probabilities, functions F such as the logistic sigmoidal or the softmax can be applied to x^L (Goodfellow, Bengio, Courville, & Bengio, 2016). The classifier $\mathcal{C}$ modeled by the DNN then classifies an input x with the label correspondent to the maximum activation $\mathcal{C}(x) = arg\,max_{i=1,\dots,n^L} F(x_i^L)$.

The task of training a DNN consists of determining the weights W^l and the biases b^l that make the model best fit the training data, according to a certain measure of training loss. In multivariate regression with K response variables (Izenman, 2008; Mielke & Berry, 1997), the training loss $\mathcal{L}$ is typically the sum-of-squared errors $\sum_{k=1}^{K}\sum_{i=1}^{n}(y_{ik} - x_k^L)^2$ where y_{ik} denotes response k corresponding to the i-th input vector. For classification with K classes, cross-entropy $-\sum_{k=1}^{K}\sum_{i=1}^{n} y_{ik}\log x_k^L$ is preferred. An effective approach to minimize $\mathcal{L}$ is by gradient descent, called *back-propagation* in this setting. Typically, one is not interested in a proven local minimum of $\mathcal{L}$, as this is likely to overfit the training dataset and yield a learning model with a high variance. Similar to the Ridge regression (see Section 2), the loss function can include regularization terms, such as a weight decay term

$$\lambda\left(\sum_{l=0}^{L-1}\sum_{i=1}^{n_l}(b_i^l)^2 + \sum_{l=0}^{L-1}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l+1}}(W_{ij}^l)^2\right),$$

or alternatively a weight elimination penalty term

$$\lambda\left(\sum_{l=0}^{L-1}\sum_{i=1}^{n_l}\frac{(b_i^l)^2}{1+(b_i^l)^2} + \sum_{l=0}^{L-1}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l+1}}\frac{(W_{ij}^l)^2}{1+(W_{ij}^l)^2}\right).$$

Weight decay limits the growth of the weights, which speeds up the training via backpropagation, and has been shown to limit overfitting (see Poggio et al., 2017 for a discussion about overfitting in Neural Networks).

The aim of this section is to present the optimization models that are used in DNN for feedforward architectures. Several other neural network architectures have been investigated in deep learning (Goodfellow et al., 2016). In particular, Convolutional Neural Networks (CNN) (LeCun et al., 1989) have been successfully adopted for processing data with a grid-like topology, such as images (Krizhevsky, Sutskever, & Hinton, 2012), videos (Karpathy et al., 2014), and traffic analytics (Wang, Zhang, Liu, Dai, & Lee, 2019). In CNN, the output of layers is obtained via convolutions (instead of the matrix multiplication in feedforward networks), and pooling operations on nearby units (such as average or maximum operators). In the remainder of the section, mixed integer programming models for DNN training are introduced in Section 6.1, and ensemble approaches with multiple activation functions are discussed in Section 6.2.

6.1. Mixed integer programming for DNN architectures

Motivated by the considerable improvements of mixed integer programming solvers, a natural question is how to model a trained DNN as a MIP. In Fischetti and Jo (2018), DNNs with *ReLU* activation

$$x^l = ReLU(W^{l-1}x^{l-1} + b^{l-1}) \quad \forall l = 1,\dots,L \tag{88}$$

are modeled as a MIP with decision variables x^l expressing the output vector of layer l, $l > 0$ and l^0 is the input vector. To express (88) explicitly, each unit $U(j, l)$ of the DNN is associated with binary activation variables z_j^l, and continuous slack variables s_j^l. The following mixed integer linear problem is proposed

$$\min \sum_{l=0}^{L}\sum_{j=1}^{n_l} c_j^l x_j^l + \sum_{l=1}^{L}\sum_{j=1}^{n_l}\gamma_j^l z_j^l \tag{89}$$

$$\text{s.t.} \sum_{i=1}^{n_{l-1}} w_{ij}^{l-1}x_i^{l-1} + b_j^{l-1} = x_j^l - s_j^l \qquad \forall l = 1,\dots,L,\ j = 1,\dots,n_l, \tag{90}$$

$$x_j^l \le (1 - z_j^l)M_x^{j,l} \qquad \forall l = 1,\dots,L,\ j = 1,\dots,n_l, \tag{91}$$

$$s_j^l \ge z_j^l M_s^{j,l} \qquad \forall l = 1,\dots,L,\ j = 1,\dots,n_l, \tag{92}$$

$$0 \le x_j^l \le ub_j^l \qquad \forall l = 1,\dots,L,\ j = 1,\dots,n_l, \tag{93}$$

$$0 \le s_j^l \le \overline{ub}_j^l \qquad \forall l = 1,\dots,L,\ j = 1,\dots,n_l, \tag{94}$$

where $M_x^{j,l}, M_s^{j,l}$ are suitably large constants. We note that since the DNN is trained, the weights w_{ij}^l and bias b_j^l are fixed parameters. Depending on the application, different activation weights c_j^l and activation costs γ_j^l can also be used for each $U(j, l)$. If known, upper bound ub_j^l can be enforced on the output x_j^l of unit $U(j, l)$ via constraints (93), and slack s_j^l can be bounded by $\overline{ub}_j^l$ via constraints (94).

The proposed MIP is feasible for every input vector x^0 since it computes the activation in the subsequent layers. Constraints (91) and (92) are known to have a weak continuous relaxation, and the tightness of the chosen constants (bounds) is crucial for their effectiveness. Several optimization solvers can directly handle such kind of constraints as indicator constraints (Bonami, Lodi, Tramontani, & Wiese, 2015). In Fischetti and Jo (2018), a bound-tightening strategy to reduce the computational times is proposed and the largest DNN tested with this approach is a 5-layer DNN with $20 + 20 + 10 + 10 + 10$ internal units.

Problem (89)–(94) can model several tasks in Deep Learning, other than the computation of quantitative responses in regression, and of classification. Such tasks include

- *Pooling operations:* The average and the maximum operators

$$Avg(x^l) = \frac{1}{n^l}\sum_{i=1}^{n^l} x_i^l,$$

$$Max(x^l) = \max(x_1^l,\dots,x_{n^l}^l),$$

can be incorporated in the hidden layers. In the case of max pooling operations, additional indicator constraints are required. For example, average and maximum operators are often used in CNNs, as mentioned earlier in Section 6.

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

- *Maximizing the unit activation:* By maximizing the objective function (89), one can find input examples x^0 that maximize the activation of the units. This may be of interest in applications such as the visualization of image features.
- *Building crafted adversarial examples:* Given an input vector x^0 labeled as χ by the DNN, the search for perturbations of x^0 that are classified as $\chi' \neq \chi$ (*adversarial examples*), can be conducted by adding conditions on the activation of the final layer L and minimizing the perturbation. In Fischetti and Jo (2018), such conditions are actually restricting the search for adversarial examples and the resulting formulation does not guarantee an adversarial solution nor can prove that no adversarial examples exist. Adversarial learning is the objective of the discussion in Section 7.
- *Training:* In this case, the weights and biases are decision variables. The resulting bilinear terms in (90) and the considerable number of decision variables in the formulation limit the applicability of (89)–(94) for DNN training.

Another attempt in modelling DNNs via MIPs is provided by Khalil, Gupta, and Dilkina (2018), in the context of Binarized Neural Networks (BNNs). BNNs are characterized by having binary weights $\{-1, +1\}$ and by using the sign function for neuron activation (Courbariaux, Bengio, & David, 2015). In Khalil et al. (2018), a MIP is proposed for finding adversarial examples in BNNs by maximizing the difference between the activation of the targeted label χ' and the predicted label χ of the input x^0, in the final layer (namely, $\max x^L_{\chi'} - x^L_{\chi}$). Contrary to Fischetti and Jo (2018), the MIP of Khalil et al. (2018) does not impose limitations on the search of adversarial examples, apart from the perturbation quantity. In terms of optimality criterion however, searching for the proven largest misclassified example is different from finding a targeted adversarial example. Furthermore, while there is interest in minimally perturbed adversarial examples, suboptimal solutions corresponding to adversarial examples (i.e., $x^L_{\chi'} \geq x^L_{\chi}$) may have a perturbation smaller than that of the optimal solution. Recently, Icarte et al. (2019) investigated a hybrid constraint programming/mixed integer programming method to train BNNs. Such model-based approach provides solutions that generalize better than those found by the largely adopted training solvers, such as gradient descent, especially for small datasets. We note that methods such as gradient descent can usually only guarantee local optimality (unless early stopping takes place).

Besides Fischetti & Jo (2018), other MIP frameworks have been proposed to model certain properties of neural networks in a bounded input domain. In Cheng, Nührenberg, and Ruess (2017), the problem of computing maximum perturbation bounds for DNNs is formulated as a MIP, where indicator constraints and disjunctive constraints are modeled using constraints with big-M coefficients (Grossmann, 2002). The maximum perturbation bound is a threshold such that the perturbed input may be classified correctly with a high probability. A restrictive misclassification condition is added when formulating the MIP. Hence, the infeasibility of the MIP does not certify the absence of adversarial examples. In addition to the *ReLU* activation, the tan^{-1} function is also considered by introducing quadratic constraints and several heuristics are proposed to solve the resulting problem. In Tjeng and Tedrake (2017), a model to formally measure the vulnerability to adversarial examples is proposed (the concept of vulnerability of neural networks is discussed in more details in Sections 7.1 and 7.2). A tight formulation for the resulting nonlinearities and a novel presolve technique are introduced to limit the number of binary variables and improve the numerical conditioning. However, the misclassification condition of adversarial examples is not explicitly defined but is rather left in the form "different from" and not explicitly modeled using equality/inequality constraints. In Serra, Tjandraatmadja, and Ramalingam (2018), the aim is to count or bound the number of linear regions that a piecewise linear classifier represented by a DNN can attain. Assuming that the input space is bounded and polyhedral, the DNN is modeled as a MIP. The contributions of adopting a MIP framework in this context are limited, especially in comparison with the computational results achieved in Montúfar (2017).

MIP frameworks can also be used to formulate the verification problem for neural networks as a satisfiability problem. In Katz, Barrett, Dill, Julian, and Kochenderfer (2017), a satisfiability modulo theory solver is proposed based on an extension of the simplex method to accommodate the *ReLU* activation functions. In Bunel, Turkaslan, Torr, Kohli, and Mudigonda (2018), a branch-and-bound framework for verifying piecewise-linear neural networks is introduced. For a recent survey on the approaches for automated verification of NNs, the reader is referred to Leofante, Narodytska, Pulina, and Tacchella (2018).

6.2. Activation ensembles

Another research direction in neural network architectures investigates the possibility of adopting multiple activation functions inside the layers of a neural network, to increase the accuracy of the classifier. Some examples in this framework are given by the *maxout* units (Goodfellow, Warde-Farley, Mirza, Courville, & Bengio, 2013), returning the maximum of multiple linear affine functions, and the *network-in-network* paradigm (Lin, Chen, & Yan, 2013) where the *ReLU* activation function is replaced by fully connected network. In Agostinelli, Hoffman, Sadowski, and Baldi (2014), adaptive piecewise linear activation functions are learned when training each neuron. Specifically, for each unit i and value z, activation $\sigma_i(z)$ is considered as

$$\sigma_i(z) = \max(0, z) + \sum_{s=1}^{S} a_i^s \max(0, -z + b_i^s), \tag{95}$$

where the number of hinges S is a hyperparameter to be fixed before training, while the variables a_i^s, b_i^s have to be learned. Functions σ_i generalize the *ReLU* function (first term of (95)), and can approximate a class of continuous piecewise-linear functions, for large enough S (Agostinelli et al., 2014).

In a more general perspective, ensemble layers are proposed in Klabjan and Harmon (2019) to consider multiple activation functions in a neural network. The idea is to embed a family of activation functions $\{\Phi^1, \ldots, \Phi^m\}$ and let the network itself choose the magnitude of their activation for each neuron i during the training. To promote relatively equal contribution to learning, the activation functions need to be scaled to the interval [0,1]. In order to measure the impact of the activation in the neural network, each function Φ^j is associated with a continuous variable α^j. The resulting activation σ_i for neuron i is then given by

$$\sigma_i(z) = \sum_{j=1}^{m} \alpha_i^j \cdot \frac{\Phi^j(z) - \min_{x \in X}(\Phi^j(z_{x,i}))}{\max_{x \in X}(\Phi^j(z_{x,i})) - \min_{x \in X}(\Phi^j(z_{x,i})) + \epsilon}, \tag{96}$$

where $z_{x,i}$ is the output of neuron i associated with training example x, X is the set of training observations, and ϵ is a small tolerance. Eq. (96) is a weighted sum of the scaled Φ^j functions, which is integrated in the training of the DNN architecture. The min and max in (96) can be approximated on a minibatch of observations in X, in the testing phase. In order to impose the selection of functions Φ^j, the magnitude of the weights α^j is limited in a projection subproblem, where for each neuron the network should choose an activation function and therefore all α^j should sum to 1. If $\hat{\alpha_j}$ are the weight values obtained by gradient descent while training, then the projected weights are found by solving the

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

convex quadratic programming problem

$$\min \sum_{j=1}^{m} \frac{1}{2}(\alpha^j - \hat{\alpha}^j)^2 \tag{97}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} \alpha^j = 1, \tag{98}$$

$$\alpha^j \geq 0 \quad \forall j = 1, \ldots, m, \tag{99}$$

which can be solved in closed form via the Karush-Kuhn-Tucker (KKT) conditions.

7. Adversarial learning

Despite the wide adoption of Machine Learning models in real-world applications, their integration into safety and security related use cases still necessitates thorough evaluation and research. A large number of contributions in the literature pointed out the dangers caused by perturbed examples, also called *adversarial examples*, causing classification errors (Biggio, Fumera, & Roli, 2010; Szegedy et al., 2013). Malicious attackers can thus exploit security falls in a general classifier. In case the attacker has a perfect knowledge of the classifier's architecture (i.e., the result of the training phase), then a *white-box* attack can be performed. *Black-box* attacks are instead performed without full information of the classifier. The interest in adversarial examples is also motivated by the transferability of the attacks to different trained models (Kurakin, Goodfellow, & Bengio, 2016; Tramèr et al., 2017). Adversarial learning then emerges as a framework to devise vulnerability attacks for classification models (Lowd & Meek, 2005).

From a mathematical perspective, such security issues have been formerly expressed via min-max approaches where the learner's and the attacker's loss functions are antagonistic (Dekel, Shamir, & Xiao, 2010; Globerson & Roweis, 2006; Lanckriet, Ghaoui, Bhattacharyya, & Jordan, 2002). Non-antagonistic losses are formulated as a Stackelberg equilibrium problem involving a bilevel optimization formulation (Brückner & Scheffer, 2011), or in a Nash equilibrium approach (Brückner, Kanzow, & Scheffer, 2012). These theoretical frameworks rely on the assumption of expressing the actual problem constraints in a game-theory setting, which is often not a viable option for real-life applications.

The search for adversarial examples can also be used to evaluate the efficiency of Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). A GAN is a minmax two-player game where a generative model G tries to reproduce the training data distribution and a discriminative model D estimates the probability of detecting samples coming from the true training distribution, rather than G. The game terminates at a saddle point, which is a minimum with respect to a player's strategy and a maximum for the other player's strategy. Discriminative networks can be affected by the presence of adversarial examples because the specific inputs to the classification networks are not considered in GANs training.

Adversarial attacks on the test set can be conducted in a targeted or untargeted fashion (Carlini & Wagner, 2017). In the targeted setup, the attacker aims to achieve a classification with a chosen target class (discussed in Section 7.1), while the untargeted misclassification is not constrained to achieve a specific class (Section 7.2). The robustness of DNNs to adversarial attacks is discussed in Section 7.3. Finally, data poisoning attacks are described in Section 7.4. While the majority of the cited papers of the present section refer to DNN applications, adversarial learning can, in general, be formulated for classifiers with quantitative classes, such as those discussed in Section 3.

7.1. Targeted attacks

Given a neural network classifier $f : \psi \subset \mathbb{R}^p \to \Upsilon$, an input $x \in \psi$ with label $y \in \Upsilon$, and a target label $y' \in \Upsilon$, a targeted attack consists of a perturbation r such that $f(x + r) = y'$. This corresponds to finding an input "close" to x, which is misclassified by f. Clearly, if the target y' coincides with y, the problem has the trivial solution $r = 0$ and no misclassification takes place.

The minimum adversarial problem for targeted attacks consists of finding a perturbation r by solving

$$\min_{r \in \mathbb{R}^p} \|r\|_2 \tag{100}$$

$$\text{s.t.} \quad f(x + r) = y', \tag{101}$$

$$x + r \in \psi. \tag{102}$$

The condition (102) ensures that the perturbed example $x + r$ belongs to the set of admissible inputs. The difficulty of solving problem (100)–(102) to optimality depends on the complexity of the classifier f, and the set ψ of feasible inputs. In general, it is computationally challenging to find an optimal solution to the problem, especially in the case of neural networks.

For classification of normalized images with binary pixel values, Szegedy et al. (2013) introduces the box-constrained approximation

$$\min_{r \in \mathbb{R}^p} c|r| + \mathcal{L}(x + r, y') \tag{103}$$

$$\text{s.t. } x + r \in [0, 1]^p, \tag{104}$$

where $\mathcal{L} : \psi \times \Upsilon \to \mathbb{R}^+$ denotes the loss function for training f (e.g., cross-entropy). The approximation is exact for convex loss functions, and can be solved via a line search algorithm on $c > 0$. For a fixed c, the formulation can be tackled by the box-constrained version of the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method (Byrd, Lu, Nocedal, & Zhu, 1995). In Gu and Rigazio (2014), c is fixed such that the perturbation is minimized on a sufficiently large subset X' of data points, and the mean prediction error rate of $f(x_i + r_i), x_i \in X'$ is greater than a threshold. In Carlini and Wagner (2017), the 2-norm in (100) is generalized to the l-norm with $l \in \{0, 2, \infty\}$ and an alternative formulation is introduced which includes functions $\mathcal{F}$ in the objective where $f(x + r) = y'$ is satisfied if and only if $\mathcal{F}(x + r) \leq 0$. The equivalent formulation is then

$$\min_{r \in \mathbb{R}^p} \|r\|_l + \Lambda \mathcal{F}(x + r) \tag{105}$$

$$\text{s.t. } x + r \in \psi, \tag{106}$$

where Λ is a constant that can be determined by binary search such that the solution r^* satisfies the condition $\mathcal{F}(x + r^*) \leq 0$. For the case where (106) are box constraints similar to (104), the authors propose strategies for applying optimization algorithms such as Adam (Kingma & Ba, 2014). Novel classes of attacks are identified for the considered metrics.

7.2. Untargeted attacks

In untargeted attacks, one searches for adversarial examples x' close to the original input x with label y for which the classified label y' of x' is different from y, without targeting a specific label for x'. Given that the only aim is misclassification, untargeted attacks are deemed less powerful than the targeted counterpart, and received less attention in the literature.

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

A mathematical formulation for finding minimum adversarial distortion for untargeted attacks is proposed in Tjeng and Tedrake (2017). Assuming that the output values of classifier f are expressed by the functions $f_{y'}$ associated with labels $y' \in \Upsilon$ (i.e., $f_{y'}$ are the scoring functions), and a distance metric d is given, then a minimum perturbation r for an untargeted attack is found by solving

$$\min_{r \in \mathbb{R}^p} d(r) \tag{107}$$

$$\text{s.t.} \quad \arg\max_{y' \in \Upsilon}\{f_{y'}(x+r)\} \neq y, \tag{108}$$

$$x + r \in \psi. \tag{109}$$

This formulation can easily accommodate targeted attacks in a set $T \not\ni y$ by replacing (108) with $\arg\max_{y'}\{f_{y'}(x+r)\} \in T$. The most commonly adopted metrics in the literature are the 1, 2, and ∞-norm, which can all be expressed with continuous variables, as shown in Tjeng and Tedrake (2017). The 2-norm makes the objective function of the outer-level optimization problem quadratic.

In order to express the logical constraint (108) in a mathematical programming formulation, problem (107)–(109) can be cast as the bilevel optimization problem

$$\min_{r \in \mathbb{R}^p, z \in \Upsilon} d(r) \tag{110}$$

$$\text{s.t.} \quad z - y \leq -\epsilon + Ms, \tag{111}$$

$$z - y \geq \epsilon - (1-s)M, \tag{112}$$

$$z \in \arg\max_{y' \in \Upsilon}\{f_{y'}(x+r)\}, \tag{113}$$

$$x + r \in \psi, \tag{114}$$

$$s \in \{0, 1\}, \tag{115}$$

where $\epsilon > 0$ is a small constant, z is a decision variable representing the classified label, M is a big-M coefficient, and s is a binary variable that enforces one of the constraints (111) and (112) which express the condition of misclassification $z \neq y$.

The complexity of the inner-level optimization problem is dependent on the scoring functions. Given that the upper-level feasibility set ψ is typically continuous and the lower-level variable y' ranges on a discrete set, the problem is in fact a continuous discrete bilevel programming problem (Fanghänel & Dempe, 2009) with convex quadratic function (Edmunds & Bard, 1992), which requires dedicated reformulations or approximations (Chen & Florian, 1995; Gümüş & Floudas, 2001; Jan & Chern, 1994).

We introduce an alternative mathematical formulation for finding untargeted adversarial examples satisfying condition (108). A perturbed input $x' = x + r$ for a sample x classified with label $y \in \Upsilon$ is an untargeted adversarial example if the classified label of x' is different from y. This condition is equivalent to

$$\exists\, y' \in \Upsilon \setminus \{y\} \text{ s.t. } f_{y'}(x') > f_y(x'). \tag{116}$$

Condition (116) is an existence condition, which can be formalized by introducing the functions $\tilde{\sigma}_{y'}(r) = ReLU(f_{y'}(x+r) - f_y(x+r))$, $y' \in \Upsilon \setminus \{y\}$, and the condition

$$\sum_{y' \in \Upsilon \setminus \{y\}} \tilde{\sigma}_{y'}(r) > \nu, \tag{117}$$

where parameter $\nu > 0$ enforces that at least one $\tilde{\sigma}_{y'}$ function has to be activated for a perturbation r. Therefore, untargeted adversarial examples can be found from formulation (107)–(109) by replacing condition (108) with the linear condition (117) and adding $|\Upsilon| - 1$ functions $\tilde{\sigma}_{y'}(r)$. The complexity of this approach depends on the scoring functions $f_{y'}$. The extra *ReLU* functions $\tilde{\sigma}$ can be expressed as a mixed integer formulation as done in problem (89)–(94).

7.3. Adversarial robustness

Another interesting line of research motivated by adversarial learning deals with adversarial training, which consists of techniques to make a neural network robust to adversarial attacks. The problem of measuring the robustness of a neural network is formalized in Bastani et al. (2016). The *pointwise robustness* evaluates if the classifier f on x is robust for "small" perturbations.

Formally, f is said to be (x, ϵ)-robust if

$$y' = y, \ \forall x' \text{ s.t. } \|x' - x\|_\infty \leq \epsilon. \tag{118}$$

Then, the pointwise robustness $\rho(f, x)$ is the minimum ϵ for which f fails to be (x, ϵ)-robust:

$$\rho(f, x) = \inf\{\epsilon \geq 0 \mid f \text{ is not } (x, \epsilon)\text{-robust}\}. \tag{119}$$

As detailed in Bastani et al. (2016), ρ is computed by expressing (119) as a constraint satisfiability problem. By imposing a bound on the perturbation, an estimation of the pointwise robustness can be performed by solving a MIP (Cheng et al., 2017).

A widely known defense technique is to augment the training data with adversarial examples; this however does not offer robustness guarantees on novel kinds of attacks. The adversarial training of neural network via robust optimization is investigated in Madry, Makelov, Schmidt, Tsipras, and Vladu (2017). In this setting, the goal is to train a neural network to be resistant to all attacks belonging to a certain class of perturbations. Particularly, the adversarial robustness with a saddle point (min-max) formulation is studied in Madry et al. (2017) which is obtained by augmenting the Empirical Risk Minimization paradigm.

Let $\theta \in \mathbb{R}^p$ be the set of model parameters to be learned, and $\mathcal{L}(\theta; x, y)$ be the loss function considered in the training phase (e.g., the cross-entropy loss) for training examples $x \in X$ and labels $y \in \Upsilon$, and let $\mathcal{S}$ be the set of allowed perturbations (e.g., an L_∞ ball). The aim is to minimize the worst expected adversarial loss on the set of inputs perturbed by $\mathcal{S}$

$$\min_\theta \mathbb{E}_{(x,y)}\left[\max_{r \in \mathcal{S}} \mathcal{L}(\theta; x+r, y)\right], \tag{120}$$

where the expectation value is computed on the distribution of the training samples. The saddle point problem (120) is viewed as the composition of an *inner maximization* and an *outer minimization* problem. The inner problem corresponds to attacking a trained neural network by means of the perturbations $\mathcal{S}$. The outer problem deals with the training of the classifier in a robust manner. The importance of formulation (120) stems both from the formalization of adversarial training and from the quantification of the robustness given by the objective function value on the chosen class of perturbations. To find solutions to (120) in a reasonable time, the structure of the local minima of the loss function can be explored.

Another robust training approach consists of optimizing the model parameters θ with respect to worst-case data (Shaham, Yamada, & Negahban, 2018). This is formalized by introducing a perturbation set $\mathcal{S}_x$ for each training example x. The aim is then to optimize

$$\min_\theta \sum_{x \in X} \max_{r \in \mathcal{S}_x} \mathcal{L}(\theta; x+r, y). \tag{121}$$

An alternating ascent and descent steps procedure can be used to solve (121) with the loss function approximated by the first-order Taylor expansion around the training points.

7.4. Data poisoning

A popular class of attacks for decreasing the training accuracy of classifiers is that of data poisoning, which was first studied for

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

SVMs (Biggio, Nelson, & Laskov, 2012). A data poisoning attack consists of hiding corrupted, altered or noisy data in the training dataset. In Steinhardt, Koh, and Liang (2017), worst-case bounds on the efficacy of a class of causative data poisoning attacks are studied. The causative attacks (Barreno, Nelson, Joseph, & Tygar, 2010) proceed as follow:

- a clean training dataset Γ_C with n data points drawn by a data-generating distribution is generated
- the attacker adds malicious examples Γ_M to Γ_C, to let the defender (learner) learn a bad model
- the defender learns model with parameters $\hat{\theta}$ from the full dataset $\Gamma = \Gamma_C \cup \Gamma_M$, reporting a test loss $\mathcal{L}(\hat{\theta})$.

Data poisoning can be viewed as a game between the attacker and the defender players, where the defender wants to minimize $\mathcal{L}(\hat{\theta})$, and the attacker seeks to maximize it. As discussed in Steinhardt et al. (2017), data sanitization defenses to limit the increase of test loss $\mathcal{L}(\hat{\theta})$ include two steps: (i) data cleaning (e.g., removing outliers which are likely to be poisoned examples), to produce a feasible dataset Γ', and (ii) minimizing a margin-based loss on the cleaned dataset $\Gamma \cap \Gamma'$. The learned model is then $\hat{\theta} = arg\,min_{\theta \in \Theta} \mathcal{L}(\theta; \Gamma \cap \Gamma')$.

Poisoning attacks can also be performed in *semi-online* or *online* fashion, where training data is processed in a streaming manner, and not in fixed batches (i.e., offline). In the semi-online context, the attacker can modify part of the training data stream so as to maximize the classification loss, and the evaluation of the objective (loss) is done only at the end of the training. In the fully-online scenario, the classifier is instead updated and evaluated during the training process. In Wang and Chaudhuri (2018), a white-box attacker's behavior in online learning for a linear classifier $w^T x$ (e.g., SVM with binary labels $y \in \{-1, +1\}$) is formulated. The attacker knows the order in which the training data is processed by the learner. The data stream S arrives in T instants ($S = \{S_1, \ldots, S_T\}$, with $S_t = (X_t, y_t)$) and the classification weights are updated using an online gradient descent algorithm (Zinkevich, 2003) such that $w_{t+1} = w_t - \eta_t(\nabla \mathcal{L}(w_t, (x_t, y_t))) + \nabla \Omega(w_t)$, where Ω is a regularization function, η_t is the step length of the iterate update, and $\mathcal{L}$ is a convex loss function. Let $\Gamma_T^{'}$ be the cleaned dataset at time T (which can be obtained, for instance, via the sphere and slab defenses), U be a given upper bound on the number of changed examples in Γ due to data sanitization, g be the attacker's objective (e.g., classification error on the test set), $|\cdot|$ be the cardinality of a set. The semi-online attacker optimization problem can then be formulated as

$$\max_{S \in \Gamma_T^{'}} g(w_T) \tag{122}$$

$$\text{s.t. } |\{S \setminus \Gamma\}| \leq U, \tag{123}$$

$$w_t = w_0 - \sum_{\tau=0}^{t-1} \eta_\tau (\nabla \mathcal{L}(\omega_\tau, S_\tau) + \nabla \mathcal{L}(w_\tau)), 1 \leq t \leq T. \tag{124}$$

Compared to the offline case, the weights w_t to be learned are a complex function of the data stream S, which makes the gradient computation more challenging and the KKT conditions do not hold. The optimization problem can be simplified by considering a convex surrogate for the objective function, given by the logistic loss. In addition, the expectation is conducted over a separate validation dataset and a label inversion procedure is implemented to cope with the multiple local maxima of the classifier function. The fully-online case can also be addressed by replacing objective (122) with $\sum_{t=1}^{T} g(w_t)$.

8. Emerging paradigms

8.1. Machine teaching

In all Machine Learning tasks discussed so far, the size of the training set of the machine learning models has been considered as a hyperparameter. The Teaching Dimension problem identifies the minimum size of a training set to correctly teach a model (Goldman & Kearns, 1995; Shinohara & Miyano, 1991). The teaching dimension of linear learners, such as Ridge regression, SVM, and logistic regression has been recently discussed in Liu and Zhu (2016). With the intent to generalize the teaching dimension problem to a variety of teaching tasks, (Zhu, 2015) and (Zhu, Singla, Zilles, & Rafferty, 2018) provide the *Machine Teaching* framework. Machine Teaching is essentially an inverse problem to Machine Learning. While in a learning task, the training dataset $\Gamma = (X, y)$ is given and the model parameters $\theta = \theta^*$ have to be determined, the role of a teacher is to let a learner approximately learn a given model θ^* by providing a proper set Γ of training examples (also called *teaching dataset* in this context). A Machine Teaching task requires to select: i) a Teaching Risk TR expressing the error of the learner, with respect to model θ^*; ii) a Teaching Cost TC expressing the convenience of the teaching dataset, from the prospective of the teacher, weighted by a regularization factor λ; iii) a learner L.

Formally, machine teaching can be cast as a bilevel optimization problem

$$\min_{\Gamma, \theta} \text{TR}(\theta) + \lambda \text{TC}(\Gamma) \tag{125}$$

$$\text{s.t. } \theta = \text{L}(\Gamma). \tag{126}$$

The upper optimization is the teacher's problem and the lower optimization $L(\Gamma)$ is the learner's machine learning problem. The teacher is aware of the learner, which could be a classifier (such as those of Section 3) or a deep neural network. Machine teaching encompasses a wide variety of applications, such as data poisoning attacks, computer tutoring systems, and adversarial training.

Problem (125)–(126) is, in general, challenging to solve. However, for certain convex learners, one can replace the lower problem by the corresponding KKT conditions, and reduce the problem to a single level formulation. The teacher is typically optimizing over a discrete space of teaching sets, hence, for some problem instances, the submodularity properties of the problem may be explored. For problems with a small teaching set, it is possible to formulate the teaching problem as a mixed integer nonlinear program. The computation of the optimal training set remains, in general, an open problem, and is especially challenging in the case where the learning algorithm does not have a closed-form solution with respect to the training set (Zhu, 2015).

The minimization of teaching cost can be directly enforced in the constrained formulation

$$\min_{\Gamma, \theta} \text{TC}(\Gamma) \tag{127}$$

$$\text{s.t. } \text{TR}(\theta) \leq \epsilon, \tag{128}$$

$$\theta = \text{L}(\Gamma), \tag{129}$$

which allows for either approximate or exact teaching. Alternatively, given a teaching budget B, the learning is performed via the constrained formulation

$$\min_{\Gamma, \theta} \text{TR}(\theta) \tag{130}$$

$$\text{s.t. } \text{TC}(\Gamma) \leq B, \tag{131}$$

$$\theta = \text{L}(\Gamma). \tag{132}$$

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

Other variants consider multiple learners to be taught by the same teacher (i.e., common teaching set). The teacher can aim to optimize for the worst learner (minimax risk), or the average learner (Bayes risk). For the teaching dimension problem, the teaching cost is the cardinality of the teaching dataset, namely its 0-norm. If the empirical minimization loss $\mathcal{L}$ is guiding the learning process, and λ is the regularization weight, then teaching dimension problem can be formulated as

$$\min_{\Gamma,\hat{\theta}} \lambda\|\Gamma\|_0 \tag{133}$$

$$\text{s.t.} \quad \|\hat{\theta} - \theta^*\|_2^2 \le \epsilon, \tag{134}$$

$$\hat{\theta} \in \text{argmin}_{\theta\in\Theta} \sum_{x\in X} \mathcal{L}(\theta; x) + \lambda\|\theta\|_2^2. \tag{135}$$

Machine teaching approaches tailored to specific learners have also been explored in the literature. In Zhu (2013), a method is proposed for the Bayesian learners, while Patil, Zhu, Kopeć, and Love (2014) focuses on Generalized Context Model learners. In Mei and Zhu (2015), the bilevel optimization of machine teaching is explored to devise optimal data poisoning attacks for a broad family of learners (i.e., SVM, logistic regression, linear regression). The attacker seeks the minimum training set poisoning to attack the learned model. By using the KKT conditions of the learner's problem, the bilevel formulation is turned into a single level optimization problem.

8.2. Empirical model learning

Empirical model learning (EML) aims to integrate machine learning models in combinatorial optimization in order to support decision-making in high-complexity systems through prescriptive analytics. This goes beyond the traditional what-if approaches where a predictive model (e.g., a simulation model) is used to estimate the parameters of an optimization model. A general framework for an EML approach is provided in Lombardi, Milano, and Bartolini (2017) and requires the following:

- A vector η of n decision variables η_i, with η_i feasible over the domain D_i.
- A mathematical encoding h of the Machine Learning model.
- A vector z of observables obtained from h.
- Logical predicates $g_j(\eta, z)$ such as mathematical programming inequalities or combinatorial restrictions in constraint programming.
- A cost function $f(\eta, z)$.

EML then solves the following optimization problem

$$\min\ f(\eta, z) \tag{136}$$

$$\text{s.t.} \quad g_j(\eta, z) \quad \forall j \in J, \tag{137}$$

$$z = h(\eta), \tag{138}$$

$$\eta_i \in D_i \quad \forall i = 1, \dots, n. \tag{139}$$

The combinatorial structure of the problem is defined by (136), (137), and (139) while (138) embeds the empirical machine learning model in the combinatorial problem. Embedding techniques for neural networks and decision trees are presented in Lombardi et al. (2017) using optimization approaches that include mixed integer nonlinear programming, constraint programming, and SAT Modulo Theories, and local search.

8.3. Bayesian network structure learning

Bayesian networks are a class of models that represent cause-effect relationships. These networks are learned by deriving the causal relationships from data. A Bayesian network is visually represented as a direct acyclic graph $G(N, E)$ where each of the nodes in N corresponds to one variable and the edges E are directional relations that indicate the cause and effect relationships among the variables. A conditional probability distribution is associated with every node/variables and along with the network structure expresses the conditional dependencies among all the variables. A main challenge in learning Bayesian networks is learning the network structure from the data. This is known as the Bayesian network structure learning problem. Finding the optimal Bayesian network structure is $\mathcal{NP}$-hard (Chickering, 1996). Mixed integer programming formulations of the Bayesian network structure learning have been proposed (Barlett & Cussens, 2013) and solved by using relaxations (Jaakkola, Sontag, Globerson, & Meila, 2010), cutting planes (Bartlett & Cussens, 2017; Campos & Ji, 2011; Cussens, 2011), and heuristics (Gasse, Aussem, & Elghazel, 2014; Yuan & Malone, 2013).

The case of learning Bayesian network structures when the width of the tree is bounded by a small constant is computationally tractable (Nie, Mauá, De Campos, & Ji, 2014; Parviainen, Farahani, & Lagergren, 2014). The bounded tree-width case is thus a restriction on the Bayesian network structure that limits the ability to represent exactly the underlying distribution of the data with the aim to achieve reasonable computational performance when computing the network structure. Following (Nie et al., 2014), to formulate the Bayesian network structure learning problem with a maximum tree-width w, the following binary variables are defined

$$p_{it} = \begin{cases} 1 & \text{if } P_{it} \text{ is the parent set of node } i \\ 0 & \text{otherwise} \end{cases}$$

where $i \in N$ and P_{it} is a parent set for node i. For each node i, the collection of parent sets is denoted as P_i and is assumed to be available (i.e., enumerated beforehand). Thus $P_{it} \in P_i$ with $t = 1, \dots, r_i$, and $r_i = |P_i|$ where $P_i \subset N$. Additional auxiliary variables $z_i \in [0, |N|]$, $v_i \in [0, |N|]$ where $|N|$ denotes the number of nodes in N, and $y_{ij} \in \{0, 1\}$ are introduced to enforce the tree-width and directed acyclic graph conditions. The problem is formulated as

$$\max \sum_{i\in N} \sum_{t=1}^{r_i} p_{it} s_i(P_{it}) \tag{140}$$

$$\text{s.t.} \quad \sum_{j\in N} y_{ij} \le w, \quad \forall i \in N, \tag{141}$$

$$(|N|+1) y_{ij} \le |N| + z_j - z_i \quad \forall i, j \in N, \tag{142}$$

$$y_{ij} + y_{ik} - y_{jk} - y_{kj} \le 1 \quad \forall i, j, k \in N, \tag{143}$$

$$\sum_{t=1}^{r_i} p_{it} = 1 \quad \forall i \in N, \tag{144}$$

$$(|N|+1) p_{it} \le |N| + v_j - v_i \quad \forall i \in N,\ \forall t = 1, \dots, r_i,\ \forall j \in P_{it}, \tag{145}$$

$$p_{it} \le y_{ij} + y_{ji} \quad \forall i \in N,\ \forall t = 1, \dots, r_i,\ \forall j \in P_{it}, \tag{146}$$

$$p_{it} \le y_{jk} + y_{kj} \quad \forall i \in N,\ \forall t = 1, \dots, r_i,\ \forall j, k \in P_{it}, \tag{147}$$

$$z_i \in [0, |N|],\ v_i \in [0, |N|],\ y_{ij} \in \{0, 1\},\ p_{it} \in \{0, 1\} \quad \forall i, j \in N, \quad \forall t = 1, \dots, r_i. \tag{148}$$

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

The objective function (140) maximizes the score of the acyclic graph where $s_i()$ is a score function that can be efficiently computed for every node $i \in N$ (Campos & Ji, 2011). Constraints (141)–(143) enforce a maximum tree-width w while constraints (144) and (145) enforce the directed acyclic graph condition. Constraints (146) and (147) enforce the relationship between the p and y variables and finally constraints (148) set the variable bounds and binary conditions. Another formulation for the bounded tree-width problem has been proposed in Parviainen et al. (2014) and includes an exponential number of constraints which are separated in a branch-and-cut framework. Both formulations however become computationally demanding as the number of features in the data set grows and with an increase in the tree-width limit. Several search heuristics have also been proposed as solution approaches (Nie, De Campos, & Ji, 2015; Nie et al., 2014; Scanagatta, Corani, de Campos, & Zaffalon, 2016).

9. Conclusions

Mathematical programming constitutes a fundamental aspect of many machine learning models where the training of these models is a large scale optimization problem. This paper surveyed a wide range of machine learning models namely regression, classification, clustering, and deep learning as well as the new emerging paradigms of machine teaching and empirical model learning. The important mathematical optimization models for expressing these machine learning models are presented and discussed. Exploiting the large scale optimization formulations and devising model specific solution approaches is an important line of research particularly benefiting from the maturity of commercial optimization software to solve the problems to optimality or to devise effective heuristics. However, as highlighted in Liang, Poggio, Rakhlin, and Stokes (2019) and Poggio et al. (2017), providing quantitative performance bounds remains an open problem. The nonlinearity of the models, the associated uncertainty of the data, as well as the scale of the problems represent some of the very important and compelling challenges to the mathematical optimization community. Furthermore, bilevel formulations play a big role in adversarial learning (Hamm & Noh, 2018), including adversarial training, data poisoning and neural network robustness.

Based on this survey, we summarize the distinctive features and the potential open machine learning problems that may benefit from the advances in computational optimization.

- **Regression.** The typical approaches to avoid overfitting and to handle uncertainty in the data include shrinkage methods and dimension reduction. These approaches can all be posed as mathematical programming models. General non-convex regularization to enforce sparsity without incurring shrinkage and bias (such as in lasso and ridge regularization) remain computationally challenging to solve to optimality. Investigating tighter relaxations and exact solution approaches continue to be an active line of research (Atamturk & Gomez, 2019).
- **Classification.** Classification problems can also be naturally formulated as optimization problems. Support vector machines in particular have been well studied in the optimization literature. Similar to regression, classifier sparsity is one important approach to avoid overfitting. Additionally, exploiting the kernel tricks is key as nonlinear separators are obtained without additional complexity. However, when posed as an optimization problem, it is still unclear how to exploit kernel tricks in sparse SVM optimization models. Another advantage to express machine learning problems as optimization problems and in particular classification problems is to account for inaccuracies in the data. Handling data uncertainty is a deeply explored field in the optimization literature and several practical approaches have been presented to handle uncertainty through robust and stochastic optimization. Such advances in the optimization literature are currently being investigated to improve over the standard approaches (Bertsimas et al., 2019).
- **Clustering.** Clustering problems are in general formulated as MINLPs that are hard to solve to optimality. The challenges include handling the non-convexity as well as the large scale instances which is a challenge even for linear variants such as the capacitated centred clustering (formulated as a binary linear model). Especially for large-scale instances, heuristics are typically devised. Exact approaches for clustering received less attention in the literature.
- **DNNs architectures as MIPs**. The advantage of mathematical programming approaches to model DNNs has only been showcased for relatively small size data sets due to the scale of the underlying optimization model. Furthermore, expressing misclassification conditions for adversarial examples in a non-restrictive manner, and handling the uncertainty in the training data are open problems in this context.
- **Adversarial learning and adversarial robustness.** Optimization models for the search for adversarial examples are important to identify and subsequently protect against novel sets of attacks. The complexity of the mathematical models in this context is highly dependent on the the classifier function. Untargeted attacks received less attention in the literature, and the mathematical programming formulation (110)–(114) has been introduced in Section 7.2. Furthermore, designing models robust to adversarial attacks is a two-player game, which can be cast as a bilevel optimization problem. The loss function adopted by the learner is one main complexity for the resulting mathematical model and solution approaches remain to be investigated.
- **Data poisoning**: Similar to adversarial robustness, defending against the poisoning of the training data is a two-player game. The case of online data retrieval is especially challenging for gradient-based algorithms as the KKT conditions do not hold.
- **Activation ensembles.** Activation ensembles seek a trade-off between the classifier accuracy and computational feasibility of training with a mathematical programming approach. Adopting activation ensembles to train large DNNs have not been investigated yet.
- **Machine teaching.** Posed as a bilevel optimization problem, one of the challenges in machine teaching is to devise computationally tractable single-level formulations that model the learner, the teaching risk, and the teaching cost. Machine teaching also generalizes a number of two-player games that are important in practice including data poisoning and adversarial training.
- **Empirical model learning.** This emerging paradigm can be seen as the bridge combining machine learning for parameter estimation and operations research for optimization. As such, theoretical and practical challenges remain to be investigated to propose prescriptive analytics models jointly combining learning and optimization in practical applications.

While this survey does not discuss numerical optimization techniques since they were recently reviewed in Bottou et al. (2018), Curtis and Scheinberg (2017), and Wright (2018), we note the fundamental role of the stochastic gradient algorithm (Robbins & Monro, 1985) and its variants on large scale machine learning. We also highlight the potential impact of machine learning on advancing the solution approaches of mathematical programming (Fischetti & Fraccaro, 2019; Fischetti, Lodi, & Zarpellon, 2019).

This survey has also focused on the learning process (loss minimization), however we note that challenging optimization problems also appear in the inference process, i.e., energy minimiza-

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

tion (see LeCun, Chopra, Hadsell, Ranzato, & Huang, 2006 for a comprehensive survey). In the inference step, the best output y^* is chosen from among all possible outputs given a certain input x such that an "energy function" is minimized. The energy function provides a measure of the goodness of a particular configuration of the input and output variables. Energy optimization constitute a common framework for machine learning where the training of a model aims at finding the optimal energy function.

A key part of most machine learning approaches is the choice of the hyperparameters of the learning model. The Hyperparameter Optimization (HPO) is usually driven by the data scientist's experience and the characteristics of the dataset and typically follows heuristic rules or cross-validation approaches. Alternatively, the HPO problem can be modeled as a box-constrained mathematical optimization problem (Diaz, Fokoue-Nkoutche, Nannicini, & Samulowitz, 2017), or as a bilevel optimization problem as discussed in Franceschi, Frasconi, Salzo, Grazzi, and Pontil (2018), Klatzer and Pock (2015), and Moore, Bergeron, and Bennett (2011), which provides theoretical convergence guarantees in addition to computational advantage. Automated approaches for HPO are also an active area of research in Machine Learning (Bergstra & Bengio, 2012; Elsken, Metzen, & Hutter, 2019; Wistuba, Rawat, & Pedapati, 2019).

Finally, since the recent widespread of machine learning to several research disciplines and in the mainstream industry can be largely attributed to the availability of data and the relatively easy to use libraries, we summarize in the online supplement the resources that may be of value for research.

Acknowledgment

We are very grateful to four anonymous referees for their valuable feedback and comments that helped improve the content and presentation of the paper. Joe Naoum-Sawaya was supported by NSERC Discovery Grant RGPIN-2017-03962 and Bissan Ghaddar was supported by NSERC Discovery Grant RGPIN-2017-04185.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2020.08.045

References

Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P. (2014). Learning activation functions to improve deep neural networks. *Technical Report* arXiv preprint 1412.6830.

Alam, M. A., Lin, H.-Y., Deng, H.-W., Calhoun, V. D., & Wang, Y.-P. (2018). A kernel machine method for detecting higher order interactions in multimodal datasets: Application to schizophrenia. *Journal of Neuroscience Methods, 309*, 161–174.

Aloise, D., Hansen, P., & Liberti, L. (2012). An improved column generation algorithm for minimum sum-of-squares clustering. *Mathematical Programming, 131*(1), 195–220.

Amaldi, E., & Coniglio, S. (2013). A distance-based point-reassignment heuristic for the k-hyperplane clustering problem. *European Journal of Operational Research, 227*(1), 22–29.

Amaldi, E., Coniglio, S., & Taccari, L. (2016). Discrete optimization methods to fit piecewise affine models to data points. *Computers & Operations Research, 75*, 214–230.

Aoki, Y., Hayami, K., Sterck, H. D., & Konagaya, A. (2014). Cluster Newton method for sampling multiple solutions of underdetermined inverse problems: application to a parameter identification problem in pharmacokinetics. *SIAM Journal on Scientific Computing, 36*(1), 14–44.

Atamturk, A., & Gomez, A. (2019). Rank-one convexification for sparse regression. *Technical Report* arXiv preprint 1901.10334.

Aytug, H. (2015). Feature selection for support vector machines using generalized Benders decomposition. *European Journal of Operational Research, 244*(1), 210–218.

Azad, M., & Moshkov, M. (2014a). Minimization of decision tree average depth for decision tables with many-valued decisions. *Procedia Computer Science, 35*, 368–377.

Azad, M., & Moshkov, M. (2014b). Minimization of decision tree depth for multi-label decision tables. In *Proceedings of the ieee international conference on granular computing* (pp. 7–12).

Azad, M., & Moshkov, M. (2015). Classification and optimization of decision trees for inconsistent decision tables represented as MVD tables. In *Proceedings of the federated conference on computer science and information systems* (pp. 31–38).

Azad, M., & Moshkov, M. (2017). Multi-stage optimization of decision and inhibitory trees for decision tables with many-valued decisions. *European Journal of Operational Research, 263*(3), 910–921.

Bagirov, A. M., & Yearwood, J. (2006). A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *European Journal of Operational Research, 170*(2), 578–596.

Barlett, M., & Cussens, J. (2013). Advances in Bayesian network learning using integer programming. In *Proceedings of the conference on uncertainty in artificial intelligence* (pp. 182–191).

Barreno, M., Nelson, B., Joseph, A. D., & Tygar, J. D. (2010). The security of machine learning. *Machine Learning, 81*(2), 121–148.

Bartlett, M., & Cussens, J. (2017). Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence, 244*, 258–271.

Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A., & Criminisi, A. (2016). Measuring neural net robustness with constraints. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 2613–2621). Curran Associates, Inc..

Baumann, P., Hochbaum, D. S., & Yang, Y. T. (2019). A comparative study of the leading machine learning techniques and two new optimization algorithms. *European Journal of Operational Research, 272*(3), 1041–1057.

Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7), 711–720.

Benati, S., & García, S. (2014). A mixed integer linear model for clustering with variable selection. *Computers & Operations Research, 43*, 280–285.

Bengio, Y., Lodi, A., & Prouvost, A. (2018). Machine learning for combinatorial optimization: a methodological tour d'Horizon. *Technical Report* arXiv preprint 1811.06128.

Bennett, K. P. (1992). Decision tree construction via linear programming. *Technical Report*. Center for Parallel Optimization, Computer Sciences Department, University of Wisconsin.

Bennett, K. P., & Blue, J. (1996). Optimal decision trees. *Technical Report*. Rensselaer Polytechnic Institute.

Bennett, K. P., & Mangasarian, O. L. (1992). Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software, 1*(1), 23–34.

Bennett, K. P., & Parrado-Hernández, E. (2006). The interplay of optimization and machine learning research. *Journal of Machine Learning Research, 7*, 1265–1281.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13*(Feb), 281–305.

Bertsimas, D., & Copenhaver, M. S. (2018). Characterization of the equivalence of robustification and regularization in linear and matrix regression. *European Journal of Operational Research, 270*(3), 931–942.

Bertsimas, D., & Dunn, J. (2017). Optimal classification trees. *Machine Learning, 106*(7), 1039–1082.

Bertsimas, D., Dunn, J., Pawlowski, C., & Zhuo, Y. D. (2019). Robust classification. *INFORMS Journal on Optimization, 1*(1), 2–34.

Bertsimas, D., & Kallus, N. (2020). From predictive to prescriptive analytics. *Management Science, 66*(3), 1025–1044.

Bertsimas, D., & King, A. (2016). OR forum–An algorithmic approach to linear regression. *Operations Research, 64*(1), 2–16.

Bertsimas, D., King, A., & Mazumder, R. (2016). Best subset selection via a modern optimization lens. *The Annals of Statistics, 44*(2), 813–852.

Bertsimas, D., & Shioda, R. (2007). Classification and regression via integer optimization. *Operations Research, 55*(2), 252–271.

Bertsimas, D., Van Parys, B., et al. (2020). Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *The Annals of Statistics, 48*(1), 300–323.

Biggio, B., Fumera, G., & Roli, F. (2010). Multiple classifier systems under attack. In *Proceedings of the international workshop on multiple classifier systems* (pp. 74–83).

Biggio, B., Nelson, B., & Laskov, P. (2012). Poisoning attacks against support vector machines. In *Proceedings of the international conference on machine learning* (pp. 1467–1474).

Blanco, V., Puerto, J., & Salmerón, R. (2018). Locating hyperplanes to fitting set of points: A general framework. *Computers & Operations Research, 95*, 172–193.

Blanquero, R., Carrizosa, E., Molero-Rıo, C., & Morales, D. R. (2018). Optimal randomized classification trees. *Technical Report*.

Blanquero, R., Carrizosa, E., Molero-Río, C., & Morales, D. R. (2020). Sparsity in optimal randomized classification trees. *European Journal of Operational Research, 284*(1), 255–272.

Bonami, P., Lodi, A., Tramontani, A., & Wiese, S. (2015). On mathematical programming with indicator constraints. *Mathematical Programming, 151*(1), 191–223.

Bonami, P., Lodi, A., & Zarpellon, G. (2018). Learning a classification of mixed-integer quadratic programming problems. In *Proceedings of the international conference on the integration of constraint programming, artificial intelligence, and operations research* (pp. 595–604).

Boţ, R. I., & Lorenz, N. (2011). Optimization problems in statistical learning: Duality and optimality conditions. *European Journal of Operational Research, 213*(2), 395–404.

Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review, 60*(2), 223–311.

Bradley, P., & Mangasarian, O. (2000). Massive data discrimination via linear support vector machines. *Optimization Methods and Software, 13*(1), 1–10.
Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. London: Chapman and Hall/CRC.
Brückner, M., Kanzow, C., & Scheffer, T. (2012). Static prediction games for adversarial learning problems. *Journal of Machine Learning Research, 13*, 2617–2654.
Brückner, M., & Scheffer, T. (2011). Stackelberg games for adversarial prediction problems. In *Proceedings of the international conference on knowledge discovery and data mining* (pp. 547–555).
Bunel, R. R., Turkaslan, I., Torr, P., Kohli, P., & Mudigonda, P. K. (2018). A unified view of piecewise linear neural network verification. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 4790–4799). Curran Associates, Inc..
Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing, 16*(5), 1190–1208.
Cafieri, S., Costa, A., & Hansen, P. (2014a). Reformulation of a model for hierarchical divisive graph modularity maximization. *Annals of Operations Research, 222*(1), 213–226.
Cafieri, S., Hansen, P., & Liberti, L. (2014b). Improving heuristics for network modularity maximization using an exact algorithm. *Discrete Applied Mathematics, 163*, 65–72.
Campos, C. P. d., & Ji, Q. (2011). Efficient structure learning of Bayesian networks using constraints. *Journal of Machine Learning Research, 12*, 663–689.
Carlini, N., & Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *Proceedings of the ieee symposium on security and privacy* (pp. 39–57).
Carrizosa, E., & Guerrero, V. (2014a). Biobjective sparse principal component analysis. *Journal of Multivariate Analysis, 132*, 151–159.
Carrizosa, E., & Guerrero, V. (2014b). rs-Sparse principal component analysis: A mixed integer nonlinear programming approach with VNS. *Computers & Operations Research, 52*, 349–354.
Carrizosa, E., Martín-Barragán, B., & Morales, D. R. (2010). Binarized support vector machines. *INFORMS Journal on Computing, 22*(1), 154–167.
Carrizosa, E., Martín-Barragán, B., & Morales, D. R. (2011). Detecting relevant variables and interactions in supervised classification. *European Journal of Operational Research, 213*(1), 260–269.
Carrizosa, E., Mladenović, N., & Todosijević, R. (2013). Variable neighborhood search for minimum sum-of-squares clustering on networks. *European Journal of Operational Research, 230*(2), 356–363.
Carrizosa, E., & Morales, D. R. (2013). Supervised classification and mathematical optimization. *Computers & Operations Research, 40*(1), 150–165.
Chan, A. B., Vasconcelos, N., & Lanckriet, G. R. G. (2007). Direct convex relaxations of sparse SVM. In *Proceedings of the international conference on machine learning* (pp. 145–153).
Chatterjee, S., & Hadi, A. S. (2015). *Regression analysis by example*. New York: John Wiley & Sons.
Chaves, A. A., & Lorena, L. A. N. (2010). Clustering search algorithm for the capacitated centered clustering problem. *Computers & Operations Research, 37*(3), 552–558.
Chen, X., Yang, J., Zhang, D., & Liang, J. (2013). Complete large margin linear discriminant analysis using mathematical programming approach. *Pattern Recognition, 46*(6), 1579–1594.
Chen, Y., & Florian, M. (1995). The nonlinear bilevel programming problem: Formulations, regularity and optimality conditions. *Optimization, 32*(3), 193–209.
Cheng, C.-H., Nührenberg, G., & Ruess, H. (2017). Maximum resilience of artificial neural networks. In D. D'Souza, & K. Narayan Kumar (Eds.), *Automated technology for verification and analysis* (pp. 251–268). Cham: Springer International Publishing.
Chickering, D. M. (1996). Learning Bayesian networks is NP-complete. In *Learning from data* (pp. 121–130). Springer.
Chikalov, I., Hussain, S., & Moshkov, M. (2018). Bi-criteria optimization of decision trees with applications to data analysis. *European Journal of Operational Research, 266*(2), 689–701.
Chouldechova, A., & Hastie, T. (2015). Generalized additive model selection. *Technical Report* arXiv preprint 1506.03850.
Chu, W., & Keerthi, S. S. (2007). Support vector ordinal regression. *Neural Computation, 19*(3), 792–815.
Claassen, G., & Hendriks, T. H. (2007). An application of special ordered sets to a periodic milk collection problem. *European Journal of Operational Research, 180*(2), 754–769.
Corne, D., Dhaenens, C., & Jourdan, L. (2012). Synergies between operations research and data mining: The emerging use of multi-objective approaches. *European Journal of Operational Research, 221*(3), 469–479.
Corrente, S., Greco, S., Kadziński, M., & Słowiński, R. (2013). Robust ordinal regression in preference learning and ranking. *Machine Learning, 93*(2-3), 381–422.
Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297.
Courbariaux, M., Bengio, Y., & David, J.-P. (2015). Binaryconnect: Training deep neural networks with binary weights during propagations. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 3123–3131). Curran Associates, Inc.
Cox, L. A., Qiu, Y., & Kuehner, W. (1989). Heuristic least-cost computation of discrete classification functions with uncertain argument values. *Annals of Operations Research, 21*(1), 1–29.
Cunningham, J. P., & Ghahramani, Z. (2015). Linear dimensionality reduction: survey, insights, and generalizations. *The Journal of Machine Learning Research, 16*(1), 2859–2900.
Curtis, F. E., & Scheinberg, K. (2017). Optimization methods for supervised machine learning: From linear models to deep learning. In *Leading developments from INFORMS communities* (pp. 89–114). INFORMS.
Cussens, J. (2011). Bayesian network learning with cutting planes. In *Proceedings of the conference on uncertainty in artificial intelligence* (pp. 153–160).
Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems, 2*(4), 303–314.
D'Ambrosio, C., Lodi, A., Wiese, S., & Bragalli, C. (2015). Mathematical programming techniques in water network optimization. *European Journal of Operational Research, 243*(3), 774–788.
Dekel, O., Shamir, O., & Xiao, L. (2010). Learning to classify with missing and corrupted features. *Machine Learning, 81*(2), 149–178.
Diaz, G. I., Fokoue-Nkoutche, A., Nannicini, G., & Samulowitz, H. (2017). An effective algorithm for hyperparameter optimization of neural networks. *IBM Journal of Research and Development, 61*(4). 9–1
Dıaz-Bánez, J., Mesa, J. A., & Schöbel, A. (2004). Continuous location of dimensional structures. *European Journal of Operational Research, 152*(1), 22–44.
Ding, C., & He, X. (2004). K-means clustering via principal component analysis. In *Proceedings of the international conference on machine learning* (p. 29).
Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *Technical Report* arXiv preprint 1702.08608.
Dreiseitl, S., & Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics, 35*(5-6), 352–359.
Dunbar, M., Murray, J. M., Cysique, L. A., Brew, B. J., & Jeyakumar, V. (2010). Simultaneous classification and feature selection via convex quadratic programming with application to HIV-associated neurocognitive disorder assessment. *European Journal of Operational Research, 206*(2), 470–478.
Edgeworth, F. Y. (1887). On observations relating to several quantities. *Hermathena, 6*(13), 279–285.
Edmunds, T. A., & Bard, J. F. (1992). An algorithm for the mixed-integer nonlinear bilevel programming problem. *Annals of Operations Research, 34*(1), 149–162.
Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004). Least angle regression. *The Annals of Statistics, 32*(2), 407–499.
Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey.. *Journal of Machine Learning Research, 20*(55), 1–21.
Fanghänel, D., & Dempe, S. (2009). Bilevel programming with discrete lower level problems. *Optimization, 58*(8), 1029–1047.
de Farias, I., Zhao, M., & Zhao, H. (2008). A special ordered set approach for optimizing a discontinuous separable piecewise linear function. *Operations Research Letters, 36*(2), 234–238.
Ferrari-Trecate, G., Muselli, M., Liberati, D., & Morari, M. (2003). A clustering technique for the identification of piecewise affine systems. *Automatica, 39*(2), 205–217.
Fischetti, M., & Fraccaro, M. (2019). Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks. *Computers & Operations Research, 106*, 289–297.
Fischetti, M., & Jo, J. (2018). Deep neural networks and mixed integer linear optimization. *Constraints, 23*(3), 296–309.
Fischetti, M., Lodi, A., & Zarpellon, G. (2019). Learning MILP resolution outcomes before reaching time-limit. In *Proceedings of the international conference on integration of constraint programming, artificial intelligence, and operations research* (pp. 275–291).
Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., & Pontil, M. (2018). Bilevel programming for hyperparameter optimization and meta-learning. *Technical Report* arXiv preprint 1806.04910.
Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning*: 1. New York, NY, USA: Springer Series in Statistics.
Fukunaga, K. (2013). *Introduction to Statistical Pattern Recognition*. Elsevier.
Ganesh, K., & Narendran, T. (2007). Cloves: A cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up. *European Journal of Operational Research, 178*(3), 699–717.
Gasse, M., Aussem, A., & Elghazel, H. (2014). A hybrid algorithm for Bayesian network structure learning with application to multi-label learning. *Expert Systems with Applications, 41*(15), 6755–6772.
Gaudioso, M., Gorgone, E., Labbé, M., & Rodríguez-Chía, A. M. (2017). Lagrangian relaxation for SVM feature selection. *Computers & Operations Research, 87*, 137–145.
Gaudreau, P., Hayami, K., Aoki, Y., Safouhi, H., & Konagaya, A. (2015). Improvements to the cluster Newton method for underdetermined inverse problems. *Journal of Computational and Applied Mathematics, 283*, 122–141.
Ghaddar, B., & Naoum-Sawaya, J. (2018). High dimensional data classification and feature selection using support vector machines. *European Journal of Operational Research, 265*(3), 993–1004.
Globerson, A., & Roweis, S. (2006). Nightmare at test time: robust learning by feature deletion. In *Proceedings of the international conference on machine learning* (pp. 353–360).
Goldman, S., & Kearns, M. (1995). On the complexity of teaching. *Journal of Computer and System Sciences, 50*(1), 20–31.
Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning*: 1. Cambridge: MIT Press.
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling,

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (pp. 2672–2680). Curran Associates, Inc.
Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout networks. In *Proceedings of the international conference on machine learning* (pp. 1319–1327).
Grossmann, I. E. (2002). Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering, 3*(3), 227–252.
Gu, S., & Rigazio, L. (2014). Towards deep neural network architectures robust to adversarial examples. *Technical Report* arXiv preprint 1412.5068.
Gümüş, Z. H., & Floudas, C. A. (2001). Global optimization of nonlinear bilevel programming problems. *Journal of Global Optimization, 20*(1), 1–31.
Günlük, O., Kalagnanam, J., Menickelly, M., & Scheinberg, K. (2018). Optimal Decision Trees for Categorical Data via Integer Programming. *Technical Report*. Optimization Online.
Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning, 46*(1-3), 389–422.
Hamm, J., & Noh, Y. (2018). K-Beam subgradient descent for minimax optimization. *Technical Report* arXiv preprint 1805.11640.
Hansen, P., & Jaumard, B. (1997). Cluster analysis and mathematical programming. *Mathematical Programming, 79*(1-3), 191–215.
Har-Peled, S., Roth, D., & Zimak, D. (2003). Constraint classification for multiclass classification and ranking. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in neural information processing systems* (pp. 809–816). MIT Press.
Hastie, T., & Tibshirani, R. (1986). Generalized additive models. *Statistical Science, 1*(3), 297–310.
Hastie, T., Tibshirani, R., & Tibshirani, R. J. (2017). Extended comparisons of best subset selection, forward stepwise selection, and the lasso. *Technical Report* arXiv preprint 1707.08692.
Herbrich, R. (2001). *Learning kernel classifiers: Theory and algorithms*. MIT Press.
Herbrich, R., Graepel, T., & Obermayer, K. (2000). In P. Bartlett, B. Schölkopf, D. Schuumans, & A. Smola (Eds.), *Large margin rank boundaries forordinal regression*. MIT Press.
Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks, 4*(2), 251–257.
Hyafil, L., & Rivest, R. L. (1976). Constructing optimal binary decision trees is NP–complete. *Information Processing Letters, 5*(1), 15–17.
Icarte, R. T., Illanes, L., Castro, M. P., Cire, A. A., McIlraith, S. A., & Beck, J. C. (2019). Training binarized neural networks using MIP and CP. In *Proceedings of the international conference on principles and practice of constraint programming*.
Izenman, A. J. (2008). Modern multivariate statistical techniques: Regression, classification and manifold learning. *Springer Texts in Statistics*: 10. Springer.
Jaakkola, T., Sontag, D., Globerson, A., & Meila, M. (2010). Learning Bayesian network structure using LP relaxations. In *Proceedings of the international conference on artificial intelligence and statistics* (pp. 358–365).
Jain, A. K., Murty, M. N. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys, 31*(3), 264–323.
James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning*: 112. Springer.
Jan, R.-H., & Chern, M.-S. (1994). Nonlinear integer bilevel programming. *European Journal of Operational Research, 72*(3), 574–587.
Jolliffe, I. (2011). Principal component analysis. In *International encyclopedia of statistical science* (pp. 1094–1096). Springer.
Karmitsa, N., Bagirov, A. M., & Taheri, S. (2017). New diagonal bundle method for clustering problems in large data sets. *European Journal of Operational Research, 263*(2), 367–379.
Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1725–1732).
Katz, G., Barrett, C., Dill, D. L., Julian, K., & Kochenderfer, M. J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proceedings of the international conference on computer aided verification* (pp. 97–117).
Kawano, S., Fujisawa, H., Takada, T., & Shiroishi, T. (2015). Sparse principal component regression with adaptive loading. *Computational Statistics & Data Analysis, 89*, 192–203.
Kelley, C. T. (1999). *Iterative methods for optimization*. Society for Industrial and Applied Mathematics.
Keshvari, A. (2018). Segmented concave least squares: A nonparametric piecewise linear regression. *European Journal of Operational Research, 266*(2), 585–594.
Khalil, E. B., Bodic, P. L., Song, L., Nemhauser, G., & Dilkina, B. (2016). Learning to branch in mixed integer programming. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 724–731).
Khalil, E. B., Dilkina, B., Nemhauser, G. L., Ahmed, S., & Shao, Y. (2017). Learning to run heuristics in tree search. In *Proceedings of the international joint conference on artificial intelligence* (pp. 659–666).
Khalil, E. B., Gupta, A., & Dilkina, B. (2018). Combinatorial attacks on binarized neural networks. *Technical Report* arXiv preprint 1810.03538.
Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *Technical Report* arXiv preprint 1412.6980.
Klabjan, D., & Harmon, M. (2019). Activation ensembles for deep neural networks. In *Proceeding of the IEEE international conference on big data* (pp. 206–214).
Klastorin, T. D. (1985). The p-median problem for cluster analysis: A comparative test using the mixture model approach. *Management Science, 31*(1), 84–95.
Klatzer, T., & Pock, T. (2015). Continuous hyper-parameter learning for support vector machines. In *Proceedings of the computer vision winter workshop* (pp. 39–47).
Kramer, S., Widmer, G., Pfahringer, B., & De Groeve, M. (2001). Prediction of ordinal classes using regression trees. *Fundamenta Informaticae, 47*(1-2), 1–13.
Kraus, M., Feuerriegel, S., & Oztekin, A. (2020). Deep learning in business analytics and operations research: models, applications and managerial implications. *European Journal of Operational Research, 281*(3), 628–641.
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (pp. 1097–1105). Curran Associates, Inc.
Kurakin, A., Goodfellow, I., & Bengio, S. (2016). Adversarial machine learning at scale. *Technical Report* arXiv preprint 1611.01236.
Kwatera, R. K., & Simeone, B. (1993). Clustering heuristics for set covering. *Annals of Operations Research, 43*(5), 295–308.
Lanckriet, G. R. G., Ghaoui, L. E., Bhattacharyya, C., & Jordan, M. I. (2002). A robust minimax approach to classification. *Journal of Machine Learning Research, 3*, 555–582.
LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., & Huang, F. (2006). In G. Bakir, T. Hofman, B. Scholkopt, A. Smola, & B. Taskar (Eds.), *A tutorial on energy-based learning*. MIT Press.
LeCun, Y., et al. (1989). Generalization and network design strategies. *Connectionism in Perspective, 19*, 143–155.
Leofante, F., Narodytska, N., Pulina, L., & Tacchella, A. (2018). Automated verification of neural networks: Advances, challenges and perspectives. *Technical Report* arXiv preprint 1805.09938.
Lewis, M., Wang, H., & Kochenberger, G. (2014). Exact solutions to the capacitated clustering problem: A comparison of two models. *Annals of Data Science, 1*(1), 15–23.
Liang, T., Poggio, T., Rakhlin, A., & Stokes, J. (2019). Fisher-Rao metric, geometry, and complexity of neural networks. In *Proceeding of the international conference on artificial intelligence and statistics* (pp. 888–896).
Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *Technical Report* arXiv preprint 1312.4400.
Liu, J., & Zhu, X. (2016). The teaching dimension of linear learners. *The Journal of Machine Learning Research, 17*(1), 5631–5655.
Lodi, A., & Zarpellon, G. (2017). On learning and branching: A survey. *TOP, 25*(2), 207–236.
Lombardi, M., Milano, M., & Bartolini, A. (2017). Empirical decision model learning. *Artificial Intelligence, 244*, 343–367.
Lowd, D., & Meek, C. (2005). Adversarial learning. In *Proceedings of the international conference on knowledge discovery in data mining* (pp. 641–647).
MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the berkeley symposium on mathematical statistics and probability* (pp. 281–297).
Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *Technical Report* arXiv preprint 1706.06083.
Mai, F., Fry, M. J., & Ohlmann, J. W. (2018). Model-based capacitated clustering with posterior regularization. *European Journal of Operational Research, 271*(2), 594–605.
Maldonado, S., Pérez, J., Weber, R., & Labbé, M. (2014). Feature selection for support vector machines via mixed integer linear programming. *Information Sciences, 279*, 163–175.
Mei, S., & Zhu, X. (2015). Using machine teaching to identify optimal training-set attacks on machine learners.. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 2871–2877).
Mielke, P. W., & Berry, K. J. (1997). Permutation-based multivariate regression analysis: The case for least sum of absolute deviations regression. *Annals of Operations Research, 74*, 259.
Miller, A. (2002). *Subset selection in regression*. Chapman and Hall/CRC.
Mišić, V. V. (2020). Optimization of tree ensembles. *Operations Research*. (In press)
Miyashiro, R., & Takano, Y. (2015). Mixed integer second-order cone programming formulations for variable selection in linear regression. *European Journal of Operational Research, 247*(3), 721–731.
Montúfar, G. (2017). Notes on the number of linear regions of deep neural networks. *Technical Report*. Max Planck Institute for Mathematics in the Sciences.
Moore, G., Bergeron, C., & Bennett, K. P. (2011). Model selection for primal SVM. *Machine Learning, 85*(1-2), 175–208.
Mortenson, M. J., Doherty, N. F., & Robinson, S. (2015). Operational research from taylorism to terabytes: A research agenda for the analytics age. *European Journal of Operational Research, 241*(3), 583–595.
Mulvey, J. M., & Crowder, H. P. (1979). Cluster analysis: An application of Lagrangian relaxation. *Management Science, 25*(4), 329–340.
Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM Journal on Computing, 24*(2), 227–234.
Negreiros, M., & Palhano, A. (2006). The capacitated centred clustering problem. *Computers & Operations Research, 33*(6), 1639–1663.
Nie, S., De Campos, C. P., & Ji, Q. (2015). Learning bounded tree-width Bayesian networks via sampling. In *Proceedings of the european conference on symbolic and quantitative approaches to reasoning and uncertainty* (pp. 387–396). Springer.
Nie, S., Mauá, D. D., De Campos, C. P., & Ji, Q. (2014). Advances in learning Bayesian networks of bounded treewidth. In *Advances in neural information processing systems* (pp. 2285–2293).
Olafsson, S., Li, X., & Wu, S. (2008). Operations research and data mining. *European Journal of Operational Research, 187*(3), 1429–1448.
Parviainen, P., Farahani, H. S., & Lagergren, J. (2014). Learning bounded tree-width

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045

Bayesian networks using integer linear programming. In *Artificial intelligence and statistics* (pp. 751–759).
Patil, K. R., Zhu, J., Kopeć, L. u., & Love, B. C. (2014). Optimal teaching for limited-capacity human learners. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 27* (pp. 2465–2473). Curran Associates, Inc.
Payne, H. J., & Meisel, W. S. (1977). An algorithm for constructing optimal binary decision trees. *IEEE Transactions on Computers, 26*(9), 905–916.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Éouard, D. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.
Piramuthu, S. (2004). Evaluating feature selection methods for learning in data mining applications. *European Journal of Operational Research, 156*(2), 483–494.
Poggio, T., Kawaguchi, K., Liao, Q., Miranda, B., Rosasco, L., Boix, X., … Mhaskar, H. (2017). Theory of Deep Learning III: Explaining the non-overfitting puzzle. *Technical Report* arXiv preprint 1801.00173.
Reris, R., & Brooks, J. P. (2015). Principal component analysis and optimization: a tutorial. *Technical Report*. Virginia Commonwealth University.
Robbins, H., & Monro, S. (1985). A stochastic approximation method. In *Herbert robbins selected papers* (pp. 102–109). Springer.
Rovatti, R., D'Ambrosio, C., Lodi, A., & Martello, S. (2014). Optimistic MILP modeling of non-linear optimization problems. *European Journal of Operational Research, 239*(1), 32–45.
Sağlam, B., Salman, F. S., Sayın, S., & Türkay, M. (2006). A mixed-integer programming approach to the clustering problem with an application in customer segmentation. *European Journal of Operational Research, 173*(3), 866–879.
Santi, É., Aloise, D., & Blanchard, S. J. (2016). A model for clustering data from heterogeneous dissimilarities. *European Journal of Operational Research, 253*(3), 659–672.
Scanagatta, M., Corani, G., de Campos, C. P., & Zaffalon, M. (2016). Learning treewidth-bounded bayesian networks with thousands of variables. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 1462–1470). Curran Associates, Inc.
Scheuerer, S., & Wendolsky, R. (2006). A scatter search heuristic for the capacitated clustering problem. *European Journal of Operational Research, 169*(2), 533–547.
Schöbel, A. (1998). Locating least-distant lines in the plane. *European Journal of Operational Research, 106*(1), 152–159.
Serra, T., Tjandraatmadja, C., & Ramalingam, S. (2018). Bounding and counting linear regions of deep neural networks. In *Proceeding of the international conference on machine learning* (pp. 4558–4566).
Shaham, U., Yamada, Y., & Negahban, S. (2018). Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing, 307*, 195–204.
Shashua, A., & Levin, A. (2003). Ranking with large margin principle: Two approaches. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in neural information processing systems* (pp. 961–968). MIT Press.
Shinohara, A., & Miyano, S. (1991). Teachability in computational learning. *New Generation Computing, 8*(4), 337–347.
Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing, 14*(3), 199–222.
Solomonoff, R. J. (1957). An inductive inference machine. In *IRE convention record, section on information theory: 2* (pp. 56–62).
Song, H., Triguero, I., & Özcan, E. (2019). A review on the self and dual interactions between machine learning and optimisation. *Progress in Artificial Intelligence, 8*(2), 143–165.
Steinhardt, J., Koh, P. W. W., & Liang, P. S. (2017). Certified defenses for data poisoning attacks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 3517–3529). Curran Associates, Inc.
Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *Technical Report* arXiv preprint 1312.6199.
Tamura, R., Kobayashi, K., Takano, Y., Miyashiro, R., Nakata, K., & Matsui, T. (2017). Best subset selection for eliminating multicollinearity. *Journal of the Operations Research Society of Japan, 60*(3), 321–336.
Tamura, R., Kobayashi, K., Takano, Y., Miyashiro, R., Nakata, K., & Matsui, T. (2019). Mixed integer quadratic optimization formulations for eliminating multicollinearity based on variance inflation factor. *Journal of Global Optimization, 73*(2), 431–446.
Taylan, P., Weber, G.-W., & Beck, A. (2007). New approaches to regression by generalized additive models and continuous optimization for modern applications in finance, science and technology. *Optimization, 56*(5-6), 675–698.
Tiwari, S., Wee, H., & Daryanto, Y. (2018). Big data analytics in supply chain management between 2010 and 2016: Insights to industries. *Computers & Industrial Engineering, 115*, 319–330.
Tjeng, V., & Tedrake, R. (2017). Evaluating robustness of neural networks with mixed integer programming. *Technical Report* arXiv preprint 1711.07356.
Toriello, A., & Vielma, J. P. (2012). Fitting piecewise linear continuous functions. *European Journal of Operational Research, 219*(1), 86–95.
Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., & McDaniel, P. (2017). Ensemble adversarial training: Attacks and defenses. *Technical Report* arXiv preprint 1705.07204.
Vapnik, V. (1998). *Statistical learning theory*: 3. New York: Wiley.
Vapnik, V. (2013). *The nature of statistical learning theory*. Springer Science & Business Media.
Verwer, S., & Zhang, Y. (2017). Learning decision trees with flexible constraints and objectives using integer optimization. In *Proceedings of the international conference on ai and or techniques in constraint programming for combinatorial optimization problems* (pp. 94–103).
Verwer, S., Zhang, Y., & Ye, Q. C. (2017). Auction optimization using regression trees and linear models as integer programs. *Artificial Intelligence, 244*, 368–395.
Vielma, J. P., Ahmed, S., & Nemhauser, G. (2010). Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research, 58*(2), 303–315.
Václavík, R., Novák, A., Šucha, P., & Hanzálek, Z. (2018). Accelerating the branch-and-price algorithm using machine learning. *European Journal of Operational Research, 271*(3), 1055–1069.
Wang, G., Gunasekaran, A., Ngai, E. W., & Papadopoulos, T. (2016). Big data analytics in logistics and supply chain management: Certain investigations for research and applications. *International Journal of Production Economics, 176*, 98–110.
Wang, H., Ding, C., & Huang, H. (2010). Multi-label linear discriminant analysis. In *Proceedings of the european conference on computer vision* (pp. 126–139).
Wang, L., Zhu, J., & Zou, H. (2006). The doubly regularized support vector machine. *Statistica Sinica, 16*(2), 589.
Wang, Y., & Chaudhuri, K. (2018). Data poisoning attacks against online learning. *Technical Report* arXiv preprint 1808.08994.
Wang, Y., Zhang, D., Liu, Y., Dai, B., & Lee, L. H. (2019). Enhancing transportation systems via deep learning: A survey. *Transportation Research Part C: Emerging Technologies, 99*, 144–163.
Wistuba, M., Rawat, A., & Pedapati, T. (2019). A survey on neural architecture search. *Technical Report* arXiv preprint arXiv:1905.01392.
Wright, S. J. (2018). Optimization algorithms for data analysis. In M. Mahoney, J. Duchi, & A. Gilbert (Eds.), *The mathematics of data* (pp. 49–98). American Mathematical Society.
Yuan, C., & Malone, B. (2013). Learning optimal Bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research, 48*, 23–65.
Zhu, J. (2013). Machine teaching for bayesian learners in the exponential family. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (pp. 1905–1913). Curran Associates, Inc.
Zhu, J., Rosset, S., Tibshirani, R., & Hastie, T. J. (2004). 1-Norm support vector machines. In S. Thrun, L. K. Saul, & B. Schölkopf (Eds.), *Advances in neural information processing systems* (pp. 49–56). MIT Press.
Zhu, X. (2015). Machine teaching: An inverse problem to machine learning and an approach toward optimal education.. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 4083–4087).
Zhu, X., Singla, A., Zilles, S., & Rafferty, A. N. (2018). An overview of machine teaching. *Technical Report* arXiv preprint 1801.05927.
Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the international conference on machine learning* (pp. 928–936).
Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67*(2), 301–320.

Please cite this article as: C. Gambella, B. Ghaddar and J. Naoum-Sawaya, Optimization problems for machine learning: A survey, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2020.08.045