Optimization Problems for Machine Learning: A Survey

Claudio Gambella [1], Bissan Ghaddar [2], Joe Naoum-Sawaya[2]

[1] IBM Research Ireland, Mulhuddart, Dublin 15, Ireland, [2] Ivey Business School, University of Western Ontario, London, Ontario N6G 0N1, Canada

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.

Contents

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 [192]. 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 numerical values that represent the parameters that define a problem, while the output, i.e. response, is a numerical value that represents the solution. Machine learning models fall into two categories: supervised and unsupervised learning [97, 126]. 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

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

where $p(x, y)$ is the probability of observing data point (x, y) [204]. 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 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. Thus attempting to find a model that perfectly fits every data point in the dataset is not desired since the predictive power of the model will be diminished when points that are far from typical are fitted. Typically the choice of f is restricted to a family of functions F such that

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

The degree of model complexity is generally dictated by the nature and size of the training data where less complex 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.

In unsupervised learning on the other hand, 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 [29, 142, 167]. Another area of research at the interface of machine learning and operations research is using machine learning to solve hard optimization problems and particularly NP-hard integer constrained optimization [40, 135, 136, 154, 208]. 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 [41, 42, 76, 215]. In fact, the widespread adoption of machine learning is in parts 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 [42, 76, 215] which described methodological developments for solving machine learning optimization problems; [20, 154] which discussed how machine learning advanced the solution approaches of mathematical programming; [70] which described the interactions between operations research and data mining; [24] which surveyed solution approaches to machine learning models cast as continuous optimization problems; and [193] which provided an overview on the various levels of interaction between optimization and machine learning. Particularly 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 [24]. It is therefore common for mathematical programming approaches to sacrifice optimality and convergence guarantees to obtain better generalization property, by adopting strategies such as

early stopping in gradient descent optimization [178].

Following this introductory section, regression models are discussed in Section 2 while classification and clustering models are presented in Sections 3 and 4, respectively. Deep learning models are presented in Section 6 while Section 8 discusses new emerging paradigms that include machine teaching and empirical model learning. Finally, conclusions are drawn in Section 9.

2 Regression Models

2.1 Linear Regression

Linear regression models are widely known approaches in supervised learning for predicting a quantitative response. The central assumption is that the dependence of the dependent variables (*feature measurements*, or *predictors*, or *input vector*) to the independent variable (real-valued *output*) is representable with a linear function (*regression function*) with a reasonable accuracy. Linear regression models have been largely adopted since the early era of statistics, and they preserve considerable interest, given their simplicity, their extensive range of applications, and the ease of interpretability which in its simplest form is the ability to explain in a humanly understandable way the role of the inputs in the outcome (see [85] for an extensive discussion on machine learning interpretability). 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). 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

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

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, 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.

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 [26], 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 prospective:

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

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 [30] 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 tasks are also investigated in [32]. 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 (i.e., multicollinearity detected by the condition number of the correlation matrix or the variance influence factor (VIF) [60]). To this end, [196] 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 [30]. Alternatively, [197] 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

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

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 [33, 31], sparse regression can be formulated as the best subset selection problem [162]

$$\min_{\beta} \frac{1}{2\gamma}\|\beta\|_2^2 + \frac{1}{2}\|y - X\beta\|_2^2 \tag{7}$$

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

where k is an upper bound on the number of predictors with a non-zero regression coefficient, i.e., the predictors to select. Problem (7)–(8) is NP-hard due to the cardinality constraint (8). The recent work of [31] demonstrated that the best subset selection can be solved using optimization techniques for values of p in the hundreds or thousands, obtaining near-optimal solutions. Specifically, by introducing the binary variables $s \in \{0, 1\}^p$, the sparse regression problem can be transformed into the mixed-integer quadratic programming (MIQP) formulation

$$\min_{\beta, s} \frac{1}{2\gamma}\|\beta\|_2^2 + \frac{1}{2}\|y - X\beta\|_2^2 \tag{9}$$

$$\text{s.t } -Ms_j \leq \beta_j \leq Ms_j \quad \forall j = 1, \ldots, p \tag{10}$$

$$\sum_{j=1}^{p} s_j \leq k \tag{11}$$

$$s \in \{0, 1\}^p \tag{12}$$

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 [69]. '. In order to limit the effect of noise in the input data and to avoid numerical issues, and hence make the model more robust, [33] introduces the Tikhonov regularization term $\frac{1}{2\Lambda}\|\beta\|_2^2$ with weight $\Lambda > 0$ into the objective function of problem (9)–(12) which is then solved using a cutting plane approach.

The task of finding a linear model to express the relationship between regressors and predictors 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 [36], locating a hyperplane $\gamma + x^T w = 0, \gamma \in \mathbb{R}, w \in \mathbb{R}^p$ to fit a set of points $x_i \in \mathbb{R}^p, i = 1, \ldots, n$, consists of finding $\hat{w}, \hat{\gamma} \in \arg\min_{w,\gamma} \phi(\epsilon(w, \gamma))$, where: $\epsilon(w, \gamma) = \epsilon_{\{x_1, \ldots, x_n\}}(w, \gamma)$ 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 [88]). If the number of points n is much smaller than the dimension p of the space, feature selection strategies can be applied [31, 164]. We note that hyperplane fitting is a variant of facility location problems [83, 186].

2.3 Regression Models Beyond Linearity

A natural extension of linear regression models is to consider nonlinear models, 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 [97, 126]. Alternatively, the Generalized Additive Models (GAMs) [118] 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

$$y = \beta_0 + \sum_{j=1}^{p} f_j(X_j). \tag{13}$$

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 [198] 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, [67] 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 [134]. Piecewise affine forms emerge as candidate models when the fitting function is known to be discontinuous [92], separable [80], or approximate to complex nonlinear expressions [79, 181, 207]. 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 $f_j : D_j \to \mathbb{R}$, 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, [4] 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 [201], however a partial knowledge of the subdomains is required. Alternatively, clustering techniques can be adopted for domain partitioning [92].

3 Classification

The task of classifying data is to decide the class membership of an unknown 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 clasification is found in [17]. This section reviews the common 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. The optimal class membership decision is to choose the class label y that maximizes the posterior distribution $P(y|x)$. Logistic Regression provides a functional form f and a parameter vector β to express $P(y|x)$ as $f(x, \beta)$. The parameters β are usually by maximum-likelihood estimation [86]. Generally, a logistic regression model calculates the class membership probability for one of the two categories in the dataset as

$$P(1|x, \beta) = \frac{1}{1 + e^{\beta_0 + \beta^\top x}}. \tag{14}$$

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 $P(1|x, \beta) = P(0|x, \beta) = 0.5$. The optimal parameter values β are obtained by maximizing the likelihood estimation $\Pi_{i=1}^{n} P(y_i|x_i, \beta)$ which is equivalent to

$$\min - \sum_{i=1}^{n} log P(y_i|x_i, \beta). \tag{15}$$

First order methods such as gradient descent as well as second order methods such as Newton's method can be applied to optimally solve problem (15).

To tune the logistic regression model (14), variable selection can be performed where only the most relevant subsets of the x variables are kept in the model. 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 (14) to further complicate the model at the risk of overfitting the training data. We note that variable selection using forward or backward elimination is a commonly used approach to do variable selection and to avoid overfitting in machine learning [97].

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 [18].

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 k-th 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 k-th 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 $q_i = G^\top x_i$ [210]. 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

$$S_w = \sum_{k=1}^K \sum_{x_i \in \pi_k} (x_i - \mu_k)(x_i - \mu_k)^\top \tag{16}$$

$$S_b = \sum_{k=1}^K n_k (\mu_k - \mu)(\mu_k - \mu)^\top. \tag{17}$$

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

$$\overline{S}_w = \sum_{k=1}^K \sum_{q_i \in \pi_k} (q_i - \overline{\mu}_k)(q_i - \overline{\mu}_k)^\top = G^\top S_w G \tag{18}$$

$$\overline{S}_b = \sum_{k=1}^K n_k (\overline{\mu}_k - \overline{\mu})(\overline{\mu}_k - \overline{\mu})^\top = G^\top S_b G. \tag{19}$$

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

$$\max \frac{|G^T S_b G|}{|G^T S_w G|}. \tag{20}$$

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 [98], 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 [62] 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.

LDA can also be applied for data with multiple labels. In the multi-label case, each data point can belong to multiple classes, which is often the case in image and video data (for example an image can be labeled "ear", "dog", "animal"). In [210] the equations of the within-class and the between-class scatters are extended to incorporate the correlation between the labels. Apart from this change, the LDA optimization problem (20) remains the same.

3.3 Decision Trees

Decision trees are classical models for making a decision or classification using splitting rules organized into 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 single-region tree, one iteratively searches for (typically univariate) cutpoint b for predictor X_j such that the tree with the two splitted regions $\{X|X_j < b\}$ and $\{X|X_j \geq b\}$ has 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 $\{X|a^T x < b\}$, where a is a vector. Another optimization criterion is a measure of purity [44], such as Gini's index in classification problems. To limit overfitting, it is possible to prune a decision tree so as to obtain subtrees minimizing, for example, cost complexity. For classification problems, [44] highlights that, given their greedy nature, the classical methods based on recursive splitting do not lead to the global optimality of the decision tree which limits the accuracy of decision trees. Since building optimal binary decision trees is known to be NP-hard [122], heuristic approaches based on mathematical programming paradigms, such as linear optimization [21], continuous optimization [22], dynamic programming [8, 10, 74, 175], have been proposed.

To find provably optimal decision trees, [27] 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 [27], the set of nodes is split into two sets, branch nodes and leaf nodes. The branch nodes $T_B = \{1, \ldots, \lfloor df \rfloor\}$ apply a linear split $a^\top x < b$ where the left branch includes the data that satisfy this split while the right branch includes the remaining data. At the leaf nodes $T_F = \{\lfloor df \rfloor + 1, \ldots, T\}$, a class prediction is made for the data that points that are included at that node. In [27], the splits that are applied at the branch nodes are restricted to a single variable with the option of not splitting a node. The mixed-integer programming formulation is

$$\min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \sum_{t \in T_B} d_t \tag{21}$$

$$\text{s.t. } L_t \geq N_t - N_{kt} - n(1 - c_{kt}), \quad k = 1, \ldots, K, \ \forall t \in T_L \tag{22}$$

$$L_t \leq N_t - N_{kt} + n c_{kt}, \quad k = 1, \ldots, K, \ \forall t \in T_L \tag{23}$$

$$N_{kt} = \frac{1}{2} \sum_{i=1}^{n} (q + Y_{ik}) z_{it}, \quad k = 1, \ldots, K, \ \forall t \in T_L \tag{24}$$

$$N_t = \sum_{i=1}^{n} z_{it}, \quad \forall t \in T_L \tag{25}$$

$$\sum_{k=1}^{K} c_{kt} = l_t, \quad \forall t \in T_L \tag{26}$$

$$\sum_{t \in T_L} z_{it} = 1, \quad i = 1, \ldots, n \tag{27}$$

$$z_{it} \leq l_t, \quad t \in T_L \tag{28}$$

$$\sum_{i=1}^{n} z_{it} \geq N_{\min} l_t, \quad t \in T_L \tag{29}$$

$$a_m^\top x_i + \epsilon \leq b_m + M_1(1 - z_{it}), \quad i = 1, \ldots, n, \ \forall t \in T_L, \ \forall m \in A_L(t), \tag{30}$$

$$a_m^\top x_i \geq b_m + M_2(1 - z_{it}), \quad i = 1, \ldots, n, \ \forall t \in T_L, \ \forall m \in A_R(t), \tag{31}$$

$$\sum_{j=1}^{p} a_{jt} = d_t, \quad \forall t \in T_B \tag{32}$$

$$0 \le b_t \le d_t, \quad \forall t \in T_B \tag{33}$$

$$d_t \le d_{p(t)}, \quad \forall t \in T_B \setminus \{1\} \tag{34}$$

$$L_t \ge 0, \quad \forall t \in T_L, \tag{35}$$

$$a_{jt} \in \{0, 1\}, \quad \forall j = 1, \ldots, p, \ \forall t \in T_B \tag{36}$$

$$d_t \in \{0, 1\}, \quad \forall t \in T_B. \tag{37}$$

The objective function (21) minimizes the normalized total misclassification loss $\frac{1}{\hat{L}} \sum_{t \in T_L} L_t$ and the decision tree complexity $\sum_{t \in T_B} d_t$ where α is a tuning parameter and $\hat{L}$ is the baseline loss obtained by predicting the most popular class from the entire dataset. Constraints (22)–(23) 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 (24) and (25), respectively, while constraints (26) indicate that each leaf node that is used (i.e. $l_t = 1$) should be assigned to a label $k = 1 \ldots K$. Constraints (27) indicate that each data point should be assigned to exactly one leaf node where $z_{it} = 1$ indicates that data point i is assigned to leaf node t. Constraints (28)–(29) 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 (30)–(31) 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 and $A_R(t)$ is the set of ancestors of t whose right branch has been followed on the path from the root node to node t. M_1 and M_2 are large numbers while ϵ is a small number to enforce the strict split $a^\top x < b$ at the left branch (see [27] for finding good values for M_1, M_2, and ϵ). Constraints (32)–(33) 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 (34), if $p(t)$, the parent of node t, does not apply a split then so is node t. Finally constraints (35)–(37) set the variable limits and binary conditions.

An alternative formulation to the optimal decision tree problem is provided in [112]. The main difference between the formulation of [112] and [27] is that the approach of [112] 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, [112] provides a strong formulation of the optimal decision tree problem thus improving the computational performance. Furthermore the formulation of [112] 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 [27] and [112] 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 [37, 205, 206]. The approach of [37] which is extended in [38] 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 with approaches such as 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 creates 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 types of 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 [163].

Furthermore, decision trees can also be used in a more general range of applications as algorithms for problem solving, data mining, and knowledge representation. In [9], 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 [11]. Recently, [66] 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 NP-hard, the authors 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 has received significant attention in the optimization literature [58, 203, 204]. 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 that separates the two classes of data points with a maximal separation margin measured as the width of the band that separates the two classes. 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 $w^\top x + \gamma = 0$ that geometrically separates the data points into the two classes such that no data point is misclassified [72]. 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

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

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

$$\max \frac{1}{\|w\|_2}$$
$$\text{s.t. } y_i(w^\top x_i + \gamma) \geq 1 \quad \forall i = 1, \ldots, n$$

which is equivalent to

$$\min \|w\|_2^2 \tag{38}$$
$$\text{s.t. } y_i(w^\top x_i + \gamma) \geq 1 \quad \forall i = 1, \ldots, n \tag{39}$$

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 and thus the soft-margin SVM which relaxes the condition of perfect separability.

3.4.2 Soft-Margin SVM

When the data is not linearly seperable, problem (38)–(39) is infeasible. Based on the error minimizing function of [23], [72] presented the soft margin SVM that introduces a slack into constraints (39) which allows the data points to be on the wrong side of the hyperplane. This slack is minimized as a proxy to minimizing the number of data points that are on the wrong side. The soft-margin SVM optimization problem is

$$\min \|w\|_2^2 + C \sum_{i=1}^{n} \xi_i \tag{40}$$
$$\text{s.t. } y_i(w^\top x_i + \gamma) \geq 1 - \xi_i \quad \forall i = 1, \ldots, n \tag{41}$$
$$\xi_i \geq 0 \quad \forall i = 1, \ldots, n. \tag{42}$$

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 (40)–(42) 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 [43].

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 [43, 218]. 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 [211, 222]. Several other approaches for dealing with sparsity in SVM have been proposed in [7, 87, 101, 103, 113, 160, 177]. The number of features can be explicitly modeled in (40)–(42) by using binary variables $z \in \{0,1\}^p$ where $z_j = 1$ indicates that feature j is selected and otherwise $z_j = 0$ [59]. A constraint limiting the number of features to a maximum desired number can be enforced resulting in the following mixed-integer quadratic problem

$$\min \|w\|_2^2 + C\sum_{i=1}^{n} \xi_i \tag{43}$$

$$\text{s.t. } y_i(w^\top x_i + \gamma) \geq 1 - \xi_i \quad \forall i = 1, \ldots, n \tag{44}$$

$$-Mz_j \leq w_j \leq Mz_j \quad \forall j = 1, \ldots, p \tag{45}$$

$$\sum_{j=1}^{p} z_j \leq r \tag{46}$$

$$z_j \in \{0,1\} \quad \forall j = 1, \ldots, p \tag{47}$$

$$\xi_i \geq 0 \quad \forall i = 1, \ldots, n. \tag{48}$$

Constraints (45) force $z_j = 1$ when feature j is used, i.e. $w_j \neq 0$ (M denotes a sufficiently large number). Constraints (46) set a limit r on the maximum number of features that can be used.

3.4.4 The Dual Problem and Kernel Tricks

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

$$\min \|w\|_2^2 + C\sum_{i=1}^{n} \xi_i \tag{49}$$

$$\text{s.t. } y_i(w^\top \phi(x_i) + \gamma) \geq 1 - \xi_i \quad \forall i = 1, \ldots, n \tag{50}$$

$$\xi_i \geq 0 \quad \forall i = 1, \ldots, n. \tag{51}$$

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 (49)–(51), the following dual problem is first obtained

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

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad \forall i = 1, \ldots, n$$

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

where α_i are the dual variables of constraints (50). 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

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

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad \forall i = 1, \ldots, n$$

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

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. The common kernel functions include polynomial $K(x_i, x_j) = (x_i^\top x_j + c)^d$

where d is the degree of the polynomial, radial basis function $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|_2^2}{\gamma}}$, and sigmoidal $K(x_i, x_j) = \tanh(\varphi x_i^\top x_j + c)$ [58, 120].

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 [55]. BSVM can be extended to capture the interactions between relevant variables in a linear problem [56]. 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, [28] proposes robust optimization models for logistic regression, decision trees, and support vector machines and shows increased accuracy over regularization most importantly without changing the complexity of the classification problem.

3.4.5 Support Vector Regression

Although as discussed earlier, support vector machines have been introduced for binary classification, its extension to regression, i.e. support vector regression, has received significant interest in the literature [191]. 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 $y \in \mathbb{R}^n$ [204]. Such a linear function may however not exist, and thus a slack from the desired tolerance is introduced and minimized similar to the soft-margin SVM. The corresponding optimization problem is

$$\min \|w\|_2^2 + C \sum_{i=1}^n (\xi_i^+ + \xi_i^-) \tag{52}$$

$$\text{s.t. } y_i - w^\top x_i - \gamma \leq \epsilon + \xi^+ \quad \forall i = 1, \dots, n \tag{53}$$

$$w^\top x_i + \gamma - y_i \leq \epsilon + \xi^- \quad \forall i = 1, \dots, n \tag{54}$$

$$\xi_i^+, \ \xi_i^- \geq 0 \quad \forall i = 1, \dots, n. \tag{55}$$

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 [191], kernel tricks can also be applied to (52)–(55) 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 [141]. An alternative is to encode the ordinal ranks into binary classifications at the expense of a large increase in the scale of the problems [116, 119]. An extension of SVM for ordinary data has been proposed in [189] and extended in [68]. 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 ith data sample of order $k \leq j$, should have a function value lower than the margin $\beta_j - 1$ while the data samples with orders $k > j$ should have a function value greater than the margin $\beta_j + 1$. The errors for violating these conditions are given by $\xi_{i,kj}^+$ and $\xi_{i,kj}^-$ respectively. Following [68], the associated SVM formulation is

$$\min \|w\|_2^2 + C \sum_{j=1}^{r-1} \left(\sum_{k=1}^{j} \sum_{i=1}^{n_k} \xi_{i,kj}^+ + \sum_{k=j+1}^{r} \sum_{i=1}^{n_k} \xi_{i,kj}^- \right)$$

$$\text{s.t. } w^\top x_{i,k} - \beta_j \leq -1 + \xi_{i,kj}^+, \quad \forall k = 1, \ldots, j, \ \forall j = 1, \ldots, r-1, \ \forall i = 1, \ldots, n^k$$

$$w^\top x_{i,k} - \beta_j \geq 1 - \xi_{i,kj}^-, \quad \forall k = j+1, \ldots, r, \ \forall j = 1, \ldots, r-1, \ \forall i = 1, \ldots, n^k.$$

As detailed in [68], 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 [71] 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 n unlabeled observations $X = \{x_1, \ldots, x_n\}$, 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., [115]). 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 [125].

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 [19, 159, 139, 168]. 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 [49, 50]. Defining the binary variables

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

and the centroid $\mu_j \in R^p$ of each cluster j, the problem of minimizing the within-cluster variance is formulated in [2] as the following mixed-integer nonlinear program

$$\min \sum_{i=1}^{n} \sum_{j=1}^{K} u_{ij} \|x_i - \mu_j\|_2^2 \tag{56}$$

$$\text{s.t.} \sum_{j=1}^{K} u_{ij} = 1, \quad \forall i = 1, \dots, n \tag{57}$$

$$\mu_j \in \mathbb{R}^p, \quad \forall j = 1, \dots, K \tag{58}$$

$$u_{ij} \in \{0, 1\}, \forall i = 1, \dots, n, \ \forall j = 1, \dots, K. \tag{59}$$

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

$$\min \sum_{i=1}^{n} \sum_{j=1}^{K} d_{ij}$$

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

$$d_{ij} \geq \|x_i - \mu_j\|_2^2 - M(1 - u_{ij}) \quad \forall i = 1, \dots, n, \ \forall j = 1, \dots, K$$

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

$$u_{ij} \in \{0, 1\}, \ d_{ij} \geq 0 \quad \forall i = 1, \dots, n, \ \forall j = 1, \dots, K.$$

Parameter M is a sufficiently large number. A solution approach based on the gradient method is proposed for problem (56)-(59) in [12]. Alternatively, a column generation approach for large-scale instances has been proposed in [2] and a bundle approach has been presented in [129]. The case where the space is not Euclidean is considered in [57]. Alternatively, [183] 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 [182] 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 [176]) is the two-step improvement procedure proposed in [157]. Starting from a sample of K points (centroids μ_j^0) in set X as initial cluster centers, 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, [157] merges clusters if their centroids are sufficiently close. Clustering is also used within heuristics for hard combinatorial problems ([145, 99]), and can be integrated in problems where the evaluation of multiple solutions is important (e.g. Cluster Newton Method [5, 102]). Cluster Newton method approximates the Jacobian in the domain covered by the cluster of points, instead of done locally by the traditional Newton's Method [133], 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 $1, \dots, K$, a mathematical formulation for CCCP is given in [169] as

$$\min \sum_{i=1}^{n} \sum_{j=1}^{K} \underline{s}_{ij} u_{ij} \tag{60}$$

$$\text{s.t} \sum_{j=1}^{K} u_{ij} = 1 \quad \forall i = 1, \dots, n, \tag{61}$$

$$\sum_{j=1}^{K} v_j \leq K \tag{62}$$

$$u_{ij} \leq v_j \quad \forall i = 1, \ldots, n, \ \forall j = 1, \ldots, V \tag{63}$$

$$\sum_{i=1}^{n} q_i u_{ij} \leq Q_j \quad \forall j = 1, \ldots, K \tag{64}$$

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

Parameter V 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 [61] while an alternative quadratic programming formulation is presented in [150]. Solution heuristics have also been proposed in [159] and [185].

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, in order 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 [3] as the following mixed-integer quadratically constraint quadratic problem

$$\min \ \sum_{i=1}^{n} \delta_i^2 \tag{65}$$

$$\text{s.t} \ \sum_{j=1}^{K} u_{ij} = 1 \quad \forall i = 1, \ldots, n \tag{66}$$

$$\delta_i \geq (w_j^T x_i - \gamma_j) - M(1 - u_{ij}) \quad \forall i = 1, \ldots, n, \ j = 1, \ldots, K \tag{67}$$

$$\delta_i \geq (-w_j^T x_i + \gamma_j) - M(1 - u_{ij}) \quad \forall i = 1, \ldots, n, \ j = 1, \ldots, K \tag{68}$$

$$\|w_j\|_2 \geq 1 \quad \forall j = 1, \ldots, K \tag{69}$$

$$\delta_i \geq 0 \quad \forall i = 1, \ldots, n \tag{70}$$

$$w_j \in \mathbb{R}^p, \gamma_j \in \mathbb{R} \quad \forall j = 1, \ldots, K \tag{71}$$

$$u_{ij} \in \{0, 1\} \quad \forall i \in 1, \ldots, n, \ \forall j = 1, \ldots, K. \tag{72}$$

Constraints (67)-(68) model the point to hyperplane distance via linear constraints. The non-convexity is due to Constraints (69). As a solution approach, a distance-based reassignment heuristic that outperforms spatial branch-and-bound solvers is proposed in [3].

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 Principle Component Analysis and Partial Least Squares, we note that other linear and nonlinear dimension reduction methods exist and an extensive survey with discussions on their benefits and shortcomings is presented in [75].

5.1 Principal Components

Principal Components Analysis (PCA) [128] constructs features with large variance based on the original set of features. 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

$$\max_{\phi^1} \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_j^1 x_{ij} \right)^2 \tag{73}$$

$$\text{s.t.} \sum_{j=1}^{p} (\phi_j^1)^2 = 1. \tag{74}$$

Problem (73)-(74) 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 [179]. One approach is to replace the L_2 norm in (73) with the L_1 norm.

An iterative heuristic approach can be used to obtain the principle 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 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

$$\max_{\phi^h} \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_j^h x_{ij} \right)^2 \tag{75}$$

$$\text{s.t.} \sum_{j=1}^{p} (\phi_j^h)^2 = 1 \tag{76}$$

$$\phi^{h^\top} S \phi^l = 0 \quad \forall l = 1, \ldots, h-1. \tag{77}$$

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 of 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 [132]. Since the identification of the principal components does not require any knowledge of the response y, PCA can be also adopted in unsupervised learning such as in the k-means clustering method (see Section 4.1, [84]). A known drawback of PCA is interpretability. To promote the sparsity of the projected components, and thus make them more interpretable, [54] formulates a Mixed-Integer Nonlinear Programming (MINLP) problem and shows that the level of sparsity can be imposed in the model, or alternatively, the variance of the principal components and their sparsity can be jointly maximized in a biobjective framework [53].

5.2 Partial Least Squares

Partial Least Squares (PLS) identifies transformed features $Z_1, \ldots, Z_H$ by taking both the predictors X and their corresponding response y into account, and is an approach that is specific to regression problems [97]. PLS is viable even for problems with a large number of features, because only one regressor has to be fitted in a simple regression model with one predictor. 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 PLS directions can be obtained by an iterative heuristic. 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 and 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 h-th direction ϕ^h can be found by solving the optimization problem

$$\max_{\phi^h} \text{Corr}(y, X\phi^h)^2 \times \text{Var}(X\phi^h) \tag{78}$$

$$\text{s.t.} \sum_{j=1}^p (\phi_j^h)^2 = 1 \tag{79}$$

$$\phi^{h^\top} S \phi^l = 0 \quad l = 1, \ldots, h-1 \tag{80}$$

where Corr() indicates the correlation matrix, Var() the variance, S the sample covariance matrix of X_j, and (80) ensures that Z_m is uncorrelated with the previous directions $Z_l = \sum_{j=1}^p \phi_j^l X_j$.

6 Deep Learning

Deep Learning received a first momentum until the 80s due to the universal approximation results [78, 121], where 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 [199], marketing, logistics [209], 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 Figure 1.

$\{0, \ldots, L\}$	*layers* indices
n^l	number of *units*, or *neurons*, in layer l
σ	element-wise activation function
$U(j, l)$	j-th 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 $x_i, i = 1, \ldots, n$ 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).

Table 1: Notation for DNN architectures.

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

$$x^l = \sigma(W^{l-1}x^{l-1} + b^{l-1}) \quad l = 1, \ldots, L. \tag{81}$$

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. In terms of activation functions, 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 option, mainly because it is easy to optimize with gradient-based methods, and tends to produce sparse networks (where not all neurons are activated). The output of the DNN is evaluated for regression or classification tasks. In the context of regression, the components of x^L can directly represent the
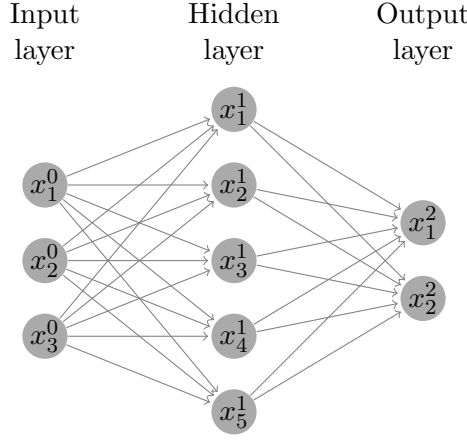
Figure 1: Deep Feedforward Neural Network with 3 layers. The input layer has $n^0 = 3$ units, hidden layer has $n^1 = 5$ units and there are $n^2 = 2$ output units. This is an example of fully connected network, where each neuron in one layer is connected to all neurons in the next layer. Training such network requires to determine weight matrices $W^0 \in \mathbb{R}^{3 \times 4}, W^1 \in \mathbb{R}^{4 \times 2}$, and bias vectors $b^1 \in \mathbb{R}^4, b^2 \in \mathbb{R}^2$.

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 [106]. The classifier $\mathcal{C}$ modeled by the DNN then classifies an input x with the label correspondent to the maximum activation: $\mathcal{C}(x) = \arg\max\limits_{i=1,\ldots,n^L} F(x_i^L)$.

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. For a regression with K quantitative responses, a common measure of training loss $\mathcal{L}$ is the sum-of-squared errors on the testing dataset $\sum\limits_{k=1}^{K}\sum\limits_{i=1}^{n}(y_{ik} - x_k^L)^2$. For classification with K classes, cross-entropy $-\sum\limits_{k=1}^{K}\sum\limits_{i=1}^{n} y_{ik} \log x_k^L$ 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 makes the training via backpropagation faster, and has been shown to limit overfitting (See [178] 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 [106]. In particular, Convolutional Neural Networks (CNN) [148] have been successfully adopted for processing data with a grid-like topology, such as images [143], videos [130], and traffic analytics [213]. 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 DNN as a MIP. In [95], DNNs with *ReLU* activation

$$x^l = ReLU(W^{l-1}x^{l-1} + b^{l-1}) \quad \forall l = 1,\ldots,L \tag{82}$$

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 (82) 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

$$\min \sum_{l=0}^{L}\sum_{j=1}^{n_l} c_j^l x_j^l + \sum_{l=1}^{L}\sum_{j=1}^{n_l} \gamma_j^l z_j^l \tag{83}$$

$$\text{s.t.} \sum_{i=1}^{n_{l-1}} w_{ij}^{l-1} x_i^{l-1} + b_j^{l-1} = x_j^l - s_j^l \qquad \forall l = 1, \ldots, L, j = 1, \ldots, n_l \tag{84}$$

$$x_j^l \leq (1 - z_j^l) M_x^{j,l} \qquad \forall l = 1, \ldots, L, j = 1, \ldots, n_l \tag{85}$$

$$s_j^l \geq z_j^l M_s^{j,l} \qquad \forall l = 1, \ldots, L, j = 1, \ldots, n_l \tag{86}$$

$$0 \leq x_j^l \leq ub_j^l \qquad \forall l = 1, \ldots, L, j = 1, \ldots, n_l \tag{87}$$

$$0 \leq s_j^l \leq \overline{ub}_j^l \qquad \forall l = 1, \ldots, L, j = 1, \ldots, n_L \tag{88}$$

where $M_x^{j,l}, M_s^{j,l}$ are suitably large constants. 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 $U(j,k)$ via constraints (87), and slack s_j^l can be bounded by $\overline{ub}_l^k$ via constraints (88).

The proposed MIP is feasible for every input vector x^0, since it computes the activation in the subsequent layers. Constraints (85) and (86) 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 [39]. In [95], 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 (83)–(88) 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

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

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

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

- *Maximizing the unit activation:* By maximizing the objective function (83), 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 [95], 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 discussed in more detail in Section 7.

- *Training:* In this case, the weights and biases are decision variables. The resulting bilinear terms in (84) and the considerable number of decision variables in the formulation limit the applicability of (83)-(88) for DNN training.

Another attempt in modelling DNNs via MIPs is provided by [137], 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 [73]. In [137], 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, $\max x_{\chi'}^L - x_\chi^L$). Contrary to [95], the MIP of [137] 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., $x_{\chi'}^L \geq x_\chi^L$) may have a perturbation smaller than that of the optimal solution. Recently, [123] investigated an 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.

Besides [95], other MIP frameworks have been proposed to model certain properties of neural networks in a bounded input domain. In [64], 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 [109]. 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 [200], 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 [187], 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 [165].

MIP frameworks can also be used to formulate the verification problem for neural networks as a satisfiability problem. In [131], a satisfiability modulo theory solver is proposed based on an extension of the simplex method to accommodate the $ReLU$ activation functions. In [47], 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 [149].

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 [108], returning the maximum of multiple linear affine functions, and the *network-in-network* paradigm [152] where the $ReLU$ activation function is replaced by fully connected network. In [1], 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

$$\sigma_i(z) = \max(0, z) + \sum_{s=1}^{S} a_i^s \max(0, -z + b_i^s), \tag{89}$$

where the number of hinges S is a hyperparameter to be fixed in advance, while the variables a_i^s, b_i^s have to be learned. Functions h_i generalize the $ReLU$ function (first term of (89)), and can approximate a class of continuous piecewise-linear functions, for large enough S [1].

In a more general perspective, Ensemble Layers are proposed in [117] 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 $[0, 1]$. 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

$$\sigma_i(z) = \sum_{j=1}^{m} \alpha_i^j \cdot \frac{\Phi^j(z) - \min_{x \in X}(\Phi^j(z_{x,i}))}{\max_{x \in X}(\Phi^j(z_{x,i})) - \min_{x \in X}(\Phi^j(z_{x,i})) + \epsilon} \tag{90}$$

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. Equation (90) is a weighted sum of the scaled Φ^j functions, which is integrated in

the training of the DNN architecture. The determination of the min and max in (90) 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 then 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 convex quadratic programming problem

$$\min_{\alpha} \ \sum_{j=1}^{m} \frac{1}{2}(\alpha^j - \hat{\alpha}^j)^2 \tag{91}$$

$$\text{s.t.} \ \sum_{j=1}^{m} \alpha^j = 1 \tag{92}$$

$$\alpha^j \geq 0, \ j = 1, \dots, m, \tag{93}$$

which can be solved in closed form via the 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 [34, 195]. 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 [144, 202]. Adversarial learning then emerges as a framework to devise vulnerability attacks for classification models [156].

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 [81, 104, 146]. Non-antagonistic losses are formulated as a Stackelberg equilibrium problem involving a bi-level optimization formulation [46], or in a Nash equilibrium approach [45]. 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) [107]. 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 that 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.

As discussed in Sections 7.1 and 7.2, adversarial attacks on the test set can be conducted in a targeted or untargeted fashion [52]. In the targeted setup, the attacker aims to achieve a classification with a chosen target class, while the untargeted misclassification is not constrained to achieve a specific class. 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 such as those discussed in Section 3.

7.1 Targeted attacks

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

In [195], the minimum adversarial problem for targeted attacks is formulated as a box-constrained problem

$$\min_{r \in \mathbb{R}^p} \ \|r\|_2 \tag{94}$$

$$\text{s.t.} \ f(x + r) = \chi' \tag{95}$$

$$x + r \in [0,1]^p. \tag{96}$$

The condition (96) ensures that the perturbed example $x + r$ belongs to the set of admissible inputs, in case of normalized images with pixel values ranging from 0 to 1. The difficulty of solving problem (94)–(96) to optimality depends on the complexity of the classifier f, and it is in general computationally challenging to find an optimal solution to the problem, especially in the case of neural networks. Denoting by $\mathcal{L} : \psi \times \Upsilon \rightarrow \mathbb{R}^+$ the loss function for training f (e.g., cross-entropy), [195] approximates the problem with a box-constrained formulation:

$$\min_{r \in \mathbb{R}^p} c|r| + \mathcal{L}(x + r, \chi') \tag{97}$$

$$x + r \in [0,1]^{n^0}. \tag{98}$$

The approximation is exact for convex loss functions, and can be solved via a line search algorithm on $c > 0$. At fixed c, the formulation can be tackled by the box-constrained version of the Limited-memory Broyden-FletcherGoldfarbShanno (L-BFGS) method [48]. In [110], 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 [52], the L_2 distance metric of formulation (94)–(96) is generalized to the l-norm with $l \in \{0, 2, \infty\}$ and an alternative formulation considers objective functions $\mathcal{F}$ satisfying $f(x + r) = \chi'$ if and only if $\mathcal{F}(x + r) \leq 0$ are introduced. The equivalent formulation is then

$$\min_{r \in \mathbb{R}^p} \|r\|_l + \Lambda \mathcal{F}(x + r, \chi') \tag{99}$$

$$x + r \in [0,1]^{n^0}, \tag{100}$$

where Λ is a constant that can be determined by binary search such that the solution r^* satisfies the condition $\mathcal{F}(x + r^*) \leq 0$. The authors propose strategies for applying optimization algorithms (such as Adam [138]) that do not support the box constraints (100) natively. Novel classes of attacks are found 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 χ for which the classified label χ' of x' is different from χ, 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.

A mathematical formulation for finding minimum adversarial distortion for untargeted attacks is proposed in [200]. Assuming that the output values of classifier f are expressed by the functions f_i associated with labels $i \in \Upsilon$ (i.e., f_i are the scoring functions), and a distance metric d is given, then a perturbation r for an untargeted attack is found by solving

$$\min_r d(r) \tag{101}$$

$$\text{s.t. } \underset{i \in \Upsilon}{\arg\max}\{f_i(x + r)\} \neq \chi \tag{102}$$

$$x + r \in \psi. \tag{103}$$

This formulation can easily accommodate targeted attacks in a set $T \not\ni \chi$ by replacing (102) with $\arg\max_i\{f_i(x + r)\} \in T$. The most commonly adopted metrics in literature are the l_1, l_2, and l_∞ norms which as shown in [200], can all be expressed with continuous variables. The 2-norm makes the objective function of the outer-level optimization problem quadratic.

In order to express the logical constraint (102) in a mathematical programming formulation, we observe that problem (101)–(103) can be cast as the bi-level optimization problem

$$\min_{r,z} d(r) \tag{104}$$

$$\text{s.t. } z - \chi \leq -\epsilon + My \tag{105}$$

$$z - \chi \geq \epsilon - (1 - y)M \tag{106}$$

$$z \in \underset{i \in \Upsilon}{\arg\max}\{f_i(x + r)\} \tag{107}$$

$$x + r \in \psi, \tag{108}$$

where $\epsilon > 0$ is a small constant and z is a decision variable representing the classified label. Constraints (105)–(106) express the condition of misclassification $z \neq \chi$ using a constraint with big-M coefficients. 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 i ranges on a discrete set, the problem is in fact a continuous discrete bilevel programming problem [91] with convex quadratic function [89], which requires dedicated reformulations or approximations [127, 63, 111].

We introduce an alternative mathematical formulation for finding untargeted adversarial examples satisfying condition (102). A perturbed input $x' = x + r$ with label χ' for the example x classified with label $\chi \in \Upsilon$ is an untargeted adversarial example if $\chi' \neq \chi$, which can be then expressed as

$$\exists \, i \in \Upsilon \setminus \{\chi\} \text{ s.t. } f_i(x') > f_\chi(x'). \tag{109}$$

Condition (109) is an existence condition, which can be formalized by introducing the functions $\tilde{\sigma}_i(r) = ReLU(f_i(x + r) - f_\chi(x))$, $i \in \Upsilon \setminus \{\chi\}$, and the condition

$$\sum_{i \in \Upsilon \setminus \{\chi\}} \tilde{\sigma}_i(r) > \nu, \tag{110}$$

where parameter $\nu > 0$ enforces that at least one $\tilde{\sigma}_i$ function has to be activated for a perturbation r. Therefore, untargeted adversarial examples can be found by modifying formulation (101)–(103) by replacing condition (102) with the linear condition (110) and adding $K-1$ functions $\tilde{\sigma}_i(r)$. The complexity of this approach depends on the scoring functions f_i. The extra $ReLU$ functions $\tilde{\sigma}$ can be expressed as a mixed-integer formulation as done in problem (83)-(88).

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 robustness of a neural network is formalized in [16]. 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

$$\chi' = \chi, \; \forall x' \text{ s.t. } \|x' - x\|_\infty \leq \epsilon. \tag{111}$$

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

$$\rho(f, x) = \inf\{\epsilon \geq 0 \mid f \text{ is not } (x, \epsilon)\text{-robust}\}. \tag{112}$$

As detailed in [16], ρ is computed by expressing (112) as a constraint satisfiability problem. By imposing a bound on the perturbation, an estimation of the pointwise robustness can be performed by casting this as a MIP [64].

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 [158]. 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 [158] 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 $\mathcal{L}(\theta; x, \chi)$ be the loss function considered in the training phase (e.g., the cross-entropy loss) for training examples $x \in X$ and labels $\chi \in \Upsilon$, 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 $\mathcal{S}$

$$\min_\theta \mathbb{E}_{(x,\chi')} \left[\max_{r \in \mathcal{S}} \mathcal{L}(\theta; x + r, \chi') \right], \tag{113}$$

where the expectation value is computed on the distribution of the training samples. The saddle point problem (113) 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 (113) 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 (113) 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 [188]. This is formalized by introducing a perturbation set $\mathcal{S}_x$ for each training example x. The aim is then to optimize

$$\min_\theta \sum_{x \in X} \max_{r \in \mathcal{S}_x} \mathcal{L}(\theta; x + r, \chi). \tag{114}$$

An alternating ascent and descent steps procedure can be used to solve (114) 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 SVMs [35]. A data poisoning attack consists of hiding corrupted, altered or noisy data in the training dataset. In [194], worst-case bounds on the efficacy of a class of causative data poisoning attacks are studied. The causative attacks [14] 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 games between the attacker and the defender players, where the defender wants to minimize $\mathcal{L}(\hat{\theta})$, and the attacker seek to maximize it. As discussed in [194], 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 $\mathcal{F}$, and (ii) minimizing a margin-based loss on the cleaned dataset $\Gamma \cap \mathcal{F}$. The learned model is then $\hat{\theta} = \arg\min_{\theta \in \Theta} \mathcal{L}(\theta; \Gamma \cap \mathcal{F})$.

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 [212], 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 [221] 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 $\mathcal{F}^T$ 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

$$\max_{S \in \mathcal{F}^T} g(w_T) \tag{115}$$

$$\text{s.t. } |\{S \setminus \Gamma\}| \leq U, \tag{116}$$

$$w_t = w_0 - \sum_{\tau=0}^{t-1} \eta_\tau (\nabla \mathcal{L}(\omega_\tau, S_\tau) + \nabla \mathcal{L}(w_\tau)), 1 \leq t \leq T. \tag{117}$$

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 Karush-Kuhn-Tucker (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 (115) 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 [105, 190]. The teaching dimension of linear learners, such as Ridge regression, SVM, and logistic regression has been recently discussed in [153]. With the intent to generalize the teaching dimension problem to a variety of teaching tasks, [219] and [220] 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

$$\min_{\Gamma,\theta} \mathrm{TR}(\theta) + \lambda \mathrm{TC}(\Gamma) \tag{118}$$

$$\text{s.t. } \theta = \mathrm{L}(\Gamma), \tag{119}$$

where 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 (118)-(119) is, in general, challenging to solve, however, for certain convex learners, one can replace the lower problem by the corresponding Karush-Kuhn-Tucker 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 of interest. 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 [219].

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

$$\min_{\Gamma,\theta} \mathrm{TC}(\Gamma) \tag{120}$$

$$\text{s.t. } \mathrm{TR}(\theta) \leq \epsilon \tag{121}$$

$$\theta = \mathrm{L}(\Gamma) \tag{122}$$

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

$$\min_{\Gamma,\theta} \mathrm{TR}(\theta) \tag{123}$$

$$\text{s.t. } \mathrm{TC}(\Gamma) \leq B \tag{124}$$

$$\theta = \mathrm{L}(\Gamma). \tag{125}$$

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

$$\min_{\Gamma,\hat{\theta}} \lambda\|\Gamma\|_0 \tag{126}$$

$$\text{s.t. } \|\hat{\theta} - \theta^*\|_2^2 \leq \epsilon \tag{127}$$

$$\hat{\theta} \in \operatorname{argmin}_{\theta \in \Theta} \sum_{x \in X} \mathcal{L}(\theta; x) + \lambda\|\theta\|_2^2. \tag{128}$$

Machine teaching approaches tailored to specific learners have also been explored in the literature. In [217], a method is proposed for the Bayesian learners, while [174] focuses on Generalized Context Model learners. In [161], 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, and solved using a gradient approach.

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 [155] 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

$$\min \ f(\eta, z) \tag{129}$$
$$\text{s.t. } g_j(\eta, z) \quad \forall j \in J \tag{130}$$
$$z = h(\eta) \tag{131}$$
$$\eta_i \in D_i \quad \forall i = 1, \dots, n. \tag{132}$$

The combinatorial structure of the problem is defined by (129), (130), and (132) while (131) embeds the empirical machine learning model in the combinatorial problem. Embedding techniques for neural networks and decision trees are presented in [155] using combinatorial 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 which is known as the Bayesian network structure learning problem. Finding the optimal Bayesian network structure is NP-hard [65]. Mixed-integer programming formulations of the Bayesian network structure learning have been proposed [13] and solved by using relaxations [124], cutting planes [15, 51, 77], and heuristics [100, 216].

The case of learning Bayesian network structures when the width of the tree is bounded by a small constant is computationally tractable [171, 173]. 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 [171], 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 $t = 1, \ldots, r_i$, 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

$$\max \sum_{i \in N} \sum_{t=1}^{r_i} p_{it} s_i(P_{it}) \tag{133}$$

$$\text{s.t.} \sum_{j \in N} y_{ij} \leq w, \quad \forall i \in N, \tag{134}$$

$$(|N| + 1)y_{ij} \leq |N| + z_j - z_i, \quad \forall i, j \in N, \tag{135}$$

$$y_{ij} + y_{ik} - y_{jk} - y_{kj} \leq 1 \quad \forall i, j, k \in N, \tag{136}$$

$$\sum_{t=1}^{r_i} p_{it} = 1 \quad \forall i \in N, \tag{137}$$

$$(|N| + 1)p_{it} \leq |N| + v_j - v_i \quad \forall i \in N, \ \forall t = 1, \ldots, r_i, \ \forall j \in P_{it}, \tag{138}$$

$$p_{it} \leq y_{ij} + y_{ji} \quad \forall i \in N, \ \forall t = 1, \ldots, r_i, \ \forall j \in P_{it}, \tag{139}$$

$$p_{it} \leq y_{jk} + y_{kj} \quad \forall i \in N, \ \forall t = 1, \ldots, r_i, \ \forall j, k \in P_{it}, \tag{140}$$

$$z_i \in [0, |N|], \ v_i \in [0, |N|], \ y_{ij} \in \{0, 1\}, \ p_{it} \in \{0, 1\} \quad \forall i, j \in N, \ \forall t = 1, \ldots, r_i. \tag{141}$$

The objective function (133) maximizes the score of the ascyclic graph where $s_i()$ is a score function that can be efficiently computed for every node $i \in N$ [51]. Constraints (134)–(136) enforce a maximum tree-width w while constraints (137)–(138) enforce the directed acyclic graph condition. Constraints (139)–(140) enforce the relationship between the p and y variables and finally constraints (141) set the variable bounds and binary conditions. Another formulation for the bounded tree-width problem has been proposed in [173] 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 [170, 172, 184].

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 [151, 178] 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 [114], 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 [6].

- **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 well studied 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 [28].

- **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 (104)–(108) 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 [42, 76, 215], we note the fundamental role of the stochastic gradient algorithm [180] 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 [93, 94].

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 minimization (see [147] 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, HPO can be modeled as a box-constrained mathematical optimization problem [82], or as a bilevel optimization problem as discussed in [96, 140, 166], which provides theoretical convergence guarantees in addition to computational advantage. Automated approaches for HPO are also an active area of research in Machine Learning [25, 90, 214].

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.

Acknowledgement

References

[1] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. Technical report, arXiv preprint 1412.6830, 2014.

[2] Daniel Aloise, Pierre Hansen, and Leo Liberti. An improved column generation algorithm for minimum sum-of-squares clustering. *Mathematical Programming*, 131(1):195–220, 2012.

[3] Edoardo Amaldi and Stefano Coniglio. A distance-based point-reassignment heuristic for the k-hyperplane clustering problem. *European Journal of Operational Research*, 227(1):22–29, 2013.

[4] Edoardo Amaldi, Stefano Coniglio, and Leonardo Taccari. Discrete optimization methods to fit piecewise affine models to data points. *Computers & Operations Research*, 75:214–230, 2016.

[5] Yasunori Aoki, Ken Hayami, Hans De Sterck, and Akihiko Konagaya. 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, 2014.

[6] Alper Atamturk and Andres Gomez. Rank-one convexification for sparse regression. Technical report, arXiv preprint 1901.10334, 2019.

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

[8] M. Azad and M. Moshkov. Minimization of decision tree depth for multi-label decision tables. In *Proceedings of the IEEE International Conference on Granular Computing (GrC)*, pages 7–12, 2014.

[9] M. Azad and M. Moshkov. 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 (FedCSIS)*, pages 31–38, 2015.

[10] Mohammad Azad and Mikhail Moshkov. Minimization of decision tree average depth for decision tables with many-valued decisions. *Procedia Computer Science*, 35:368–377, 2014.

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

[12] Adil M Bagirov and John Yearwood. A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *European Journal of Operational Research*, 170(2):578–596, 2006.

[13] Mark Barlett and James Cussens. Advances in Bayesian network learning using integer programming. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 182–191, 2013.

[14] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. Doug Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[15] Mark Bartlett and James Cussens. Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271, 2017.

[16] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2613–2621. Curran Associates, Inc., 2016.

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

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

[19] Stefano Benati and Sergio García. A mixed integer linear model for clustering with variable selection. *Computers & operations research*, 43:280–285, 2014.

[20] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. Technical report, arXiv preprint 1811.06128, 2018.

[21] Kristin P Bennett. Decision tree construction via linear programming. Technical report, Center for Parallel Optimization, Computer Sciences Department, University of Wisconsin, 1992.

[22] Kristin P. Bennett and J. Blue. Optimal decision trees. Technical report, Rensselaer Polytechnic Institute, 1996.

[23] Kristin P Bennett and Olvi L Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization methods and software*, 1(1):23–34, 1992.

[24] Kristin P Bennett and Emilio Parrado-Hernández. The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, 7:1265–1281, 2006.

[25] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[26] Dimitris Bertsimas and Martin S. Copenhaver. Characterization of the equivalence of robustification and regularization in linear and matrix regression. *European Journal of Operational Research*, 270(3):931–942, 2018.

[27] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[28] Dimitris Bertsimas, Jack Dunn, Colin Pawlowski, and Ying Daisy Zhuo. Robust classification. *INFORMS Journal on Optimization*, 1(1):2–34, 2019.

[29] Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. Technical report, arXiv preprint 1402.5481, 2014.

[30] Dimitris Bertsimas and Angela King. OR forum–An algorithmic approach to linear regression. *Operations Research*, 64(1):2–16, 2016.

[31] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *The annals of statistics*, 44(2):813–852, 2016.

[32] Dimitris Bertsimas and Romy Shioda. Classification and regression via integer optimization. *Operations Research*, 55(2):252–271, 2007.

[33] Dimitris Bertsimas and Bart Van Parys. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. Technical report, arXiv preprint 1709.10029, 2017.

[34] Battista Biggio, Giorgio Fumera, and Fabio Roli. Multiple classifier systems under attack. In *Proceedings of the International Workshop on Multiple Classifier Systems*, pages 74–83, 2010.

[35] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 1467–1474, 2012.

[36] Víctor Blanco, Justo Puerto, and Román Salmerón. Locating hyperplanes to fitting set of points: A general framework. *Computers & Operations Research*, 95:172–193, 2018.

[37] Rafael Blanquero, Emilio Carrizosa, Cristina Molero-Río, and Dolores Romero Morales. Optimal randomized classification trees. Technical report, 2018.

[38] Rafael Blanquero, Emilio Carrizosa, Cristina Molero-Río, and Dolores Romero Morales. Sparsity in optimal randomized classification trees. Technical report, 2018.

[39] Pierre Bonami, Andrea Lodi, Andrea Tramontani, and Sven Wiese. On mathematical programming with indicator constraints. *Mathematical Programming*, 151(1):191–223, 2015.

[40] Pierre Bonami, Andrea Lodi, and Giulia Zarpellon. 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*, pages 595–604, 2018.

[41] Radu Ioan Boţ and Nicole Lorenz. Optimization problems in statistical learning: Duality and optimality conditions. *European Journal of Operational Research*, 213(2):395–404, 2011.

[42] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[43] Paul Bradley and Olvi Mangasarian. Massive data discrimination via linear support vector machines. *Optimization Methods and Software*, 13(1):1–10, 2000.

[44] L Breiman, J Friedman, R Olshen, and C Stone. *Classification and regression trees*. Chapman and Hall/CRC, London, 1984.

[45] Michael Brückner, Christian Kanzow, and Tobias Scheffer. Static prediction games for adversarial learning problems. *Journal of Machine Learning Research*, 13:2617–2654, 2012.

[46] Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 547–555, 2011.

[47] Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan K Mudigonda. A unified view of piecewise linear neural network verification. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4790–4799. Curran Associates, Inc., 2018.

[48] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[49] Sonia Cafieri, Alberto Costa, and Pierre Hansen. Reformulation of a model for hierarchical divisive graph modularity maximization. *Annals of Operations Research*, 222(1):213–226, 2014.

[50] Sonia Cafieri, Pierre Hansen, and Leo Liberti. Improving heuristics for network modularity maximization using an exact algorithm. *Discrete Applied Mathematics*, 163:65–72, 2014.

[51] Cassio P de Campos and Qiang Ji. Efficient structure learning of Bayesian networks using constraints. *Journal of Machine Learning Research*, 12:663–689, 2011.

[52] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 39–57, 2017.

[53] Emilio Carrizosa and Vanesa Guerrero. Biobjective sparse principal component analysis. *Journal of Multivariate Analysis*, 132:151–159, 2014.

[54] Emilio Carrizosa and Vanesa Guerrero. rs-sparse principal component analysis: A mixed integer nonlinear programming approach with VNS. *Computers & Operations Research*, 52:349–354, 2014.

[55] Emilio Carrizosa, Beln Martn-Barragn, and Dolores Romero Morales. Binarized support vector machines. *INFORMS Journal on Computing*, 22(1):154–167, 2010.

[56] Emilio Carrizosa, Beln Martn-Barragn, and Dolores Romero Morales. Detecting relevant variables and interactions in supervised classification. *European Journal of Operational Research*, 213(1):260–269, 2011.

[57] Emilio Carrizosa, Nenad Mladenovi, and Raca Todosijevi. Variable neighborhood search for minimum sum-of-squares clustering on networks. *European Journal of Operational Research*, 230(2):356–363, 2013.

[58] Emilio Carrizosa and Dolores Romero Morales. Supervised classification and mathematical optimization. *Computers & Operations Research*, 40(1):150–165, 2013.

[59] Antoni B. Chan, Nuno Vasconcelos, and Gert R. G. Lanckriet. Direct convex relaxations of sparse SVM. In *Proceedings of the International Conference on Machine Learning*, pages 145–153, 2007.

[60] Samprit Chatterjee and Ali S Hadi. *Regression analysis by example*. John Wiley & Sons, New York, 2015.

[61] Antonio Augusto Chaves and Luiz Antonio Nogueira Lorena. Clustering search algorithm for the capacitated centered clustering problem. *Computers & Operations Research*, 37(3):552–558, 2010.

[62] Xiaobo Chen, Jian Yang, David Zhang, and Jun Liang. Complete large margin linear discriminant analysis using mathematical programming approach. *Pattern recognition*, 46(6):1579–1594, 2013.

[63] Yang Chen and Michael Florian. The nonlinear bilevel programming problem: Formulations, regularity and optimality conditions. *Optimization*, 32(3):193–209, 1995.

[64] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In Deepak D'Souza and K. Narayan Kumar, editors, *Automated Technology for Verification and Analysis*, pages 251–268, Cham, 2017. Springer International Publishing.

[65] David Maxwell Chickering. Learning Bayesian networks is NP-complete. In *Learning from data*, pages 121–130. Springer, 1996.

[66] Igor Chikalov, Shahid Hussain, and Mikhail Moshkov. Bi-criteria optimization of decision trees with applications to data analysis. *European Journal of Operational Research*, 266(2):689–701, 2018.

[67] Alexandra Chouldechova and Trevor Hastie. Generalized additive model selection. Technical report, arXiv preprint 1506.03850, 2015.

[68] Wei Chu and S Sathiya Keerthi. Support vector ordinal regression. *Neural computation*, 19(3):792–815, 2007.

[69] GDH Claassen and Th HB Hendriks. An application of special ordered sets to a periodic milk collection problem. *European Journal of Operational Research*, 180(2):754–769, 2007.

[70] David Corne, Clarisse Dhaenens, and Laetitia Jourdan. Synergies between operations research and data mining: The emerging use of multi-objective approaches. *European Journal of Operational Research*, 221(3):469–479, 2012.

[71] Salvatore Corrente, Salvatore Greco, Miłosz Kadziński, and Roman Słowiński. Robust ordinal regression in preference learning and ranking. *Machine Learning*, 93(2-3):381–422, 2013.

[72] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[73] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3123–3131. Curran Associates, Inc., 2015.

[74] Louis Anthony Cox, Yuping Qiu, and Warren Kuehner. Heuristic least-cost computation of discrete classification functions with uncertain argument values. *Annals of Operations research*, 21(1):1–29, 1989.

[75] John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research*, 16(1):2859–2900, 2015.

[76] Frank E. Curtis and Katya Scheinberg. Optimization methods for supervised machine learning: From linear models to deep learning. In *Leading Developments from INFORMS Communities*, pages 89–114. INFORMS, 2017.

[77] James Cussens. Bayesian network learning with cutting planes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 153–160, 2011.

[78] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[79] Claudia D'Ambrosio, Andrea Lodi, Sven Wiese, and Cristiana Bragalli. Mathematical programming techniques in water network optimization. *European Journal of Operational Research*, 243(3):774–788, 2015.

[80] I.R. de Farias, M. Zhao, and H. Zhao. A special ordered set approach for optimizing a discontinuous separable piecewise linear function. *Operations Research Letters*, 36(2):234–238, 2008.

[81] Ofer Dekel, Ohad Shamir, and Lin Xiao. Learning to classify with missing and corrupted features. *Machine learning*, 81(2):149–178, 2010.

[82] Gonzalo I. Diaz, Achille Fokoue-Nkoutche, Giacomo Nannicini, and Horst Samulowitz. An effective algorithm for hyperparameter optimization of neural networks. *IBM Journal of Research and Development*, 61(4):9–1, 2017.

[83] JM Dıaz-Bánez, Juan A Mesa, and Anita Schöbel. Continuous location of dimensional structures. *European Journal of Operational Research*, 152(1):22–44, 2004.

[84] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the international conference on Machine learning*, page 29, 2004.

[85] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. Technical report, arXiv preprint 1702.08608, 2017.

[86] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5-6):352–359, 2002.

[87] Michelle Dunbar, John M. Murray, Lucette A. Cysique, Bruce J. Brew, and Vaithilingam Jeyakumar. 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, 2010.

[88] Francis Y Edgeworth. On observations relating to several quantities. *Hermathena*, 6(13):279–285, 1887.

[89] Thomas A Edmunds and Jonathan F Bard. An algorithm for the mixed-integer nonlinear bilevel programming problem. *Annals of Operations Research*, 34(1):149–162, 1992.

[90] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[91] Diana Fanghänel and Stephan Dempe. Bilevel programming with discrete lower level problems. *Optimization*, 58(8):1029–1047, 2009.

[92] Giancarlo Ferrari-Trecate, Marco Muselli, Diego Liberati, and Manfred Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, 2003.

[93] Martina Fischetti and Marco Fraccaro. Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks. *Computers & Operations Research*, 106:289–297, 2019.

[94] Martina Fischetti, Andrea Lodi, and Giulia Zarpellon. Learning MILP resolution outcomes before reaching time-limit. In *Proceedings of the International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 275–291, 2019.

[95] Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, 2018.

[96] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimilano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. Technical report, arXiv preprint 1806.04910, 2018.

[97] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA, 2001.

[98] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[99] K Ganesh and TT Narendran. 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, 2007.

[100] Maxime Gasse, Alex Aussem, and Haytham Elghazel. A hybrid algorithm for Bayesian network structure learning with application to multi-label learning. *Expert Systems with Applications*, 41(15):6755–6772, 2014.

[101] Manlio Gaudioso, Enrico Gorgone, Martine Labbé, and Antonio M Rodríguez-Chía. Lagrangian relaxation for SVM feature selection. *Computers & Operations Research*, 87:137–145, 2017.

[102] Philippe Gaudreau, Ken Hayami, Yasunori Aoki, Hassan Safouhi, and Akihiko Konagaya. Improvements to the cluster Newton method for underdetermined inverse problems. *Journal of Computational and Applied Mathematics*, 283:122–141, 2015.

[103] Bissan Ghaddar and Joe Naoum-Sawaya. High dimensional data classification and feature selection using support vector machines. *European Journal of Operational Research*, 265(3):993–1004, 2018.

[104] Amir Globerson and Sam Roweis. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the international conference on Machine learning*, pages 353–360, 2006.

[105] Sally Goldman and Michael Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

[106] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[107] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[108] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *Proceedings of the International Conference on Machine Learning*, pages 1319–1327, 2013.

[109] Ignacio E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and engineering*, 3(3):227–252, 2002.

[110] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. Technical report, arXiv preprint 1412.5068, 2014.

[111] Zeynep H Gümüş and Christodoulos A Floudas. Global optimization of nonlinear bilevel programming problems. *Journal of Global Optimization*, 20(1):1–31, 2001.

[112] Oktay Günlük, Jayant Kalagnanam, Matt Menickelly, and Katya Scheinberg. Optimal decision trees for categorical data via integer programming. Technical report, Optimization Online, 2018.

[113] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[114] Jihun Hamm and Yung-Kyun Noh. K-beam subgradient descent for minimax optimization. Technical report, arXiv preprint 1805.11640, 2018.

[115] Pierre Hansen and Brigitte Jaumard. Cluster analysis and mathematical programming. *Mathematical programming*, 79(1-3):191–215, 1997.

[116] Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification for multiclass classification and ranking. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 809–816. MIT Press, 2003.

[117] Mark Harmon and Diego Klabjan. Activation Ensembles for Deep Neural Networks. Technical report, arXiv preprint 1702.07790, 2017.

[118] Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical Science*, 1(3):297–310, 1986.

[119] R. Herbrich, T. Graepel, and K. Obermayer. *Large margin rank boundaries forordinal regression*. MIT Press, 2000.

[120] Ralf Herbrich. *Learning kernel classifiers: theory and algorithms*. MIT Press, 2001.

[121] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[122] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[123] Rodrigo Toro Icarte, León Illanes, Margarita P Castro, Andre A Cire, Sheila A McIlraith, and J Christopher Beck. Training binarized neural networks using MIP and CP. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 2019.

[124] Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using lp relaxations. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 358–365, 2010.

[125] Anil K. Jain, M. N. Narasimha Murty, and Patrick J. Flynn. Data clustering: a review. *ACM computing surveys*, 31(3):264–323, 1999.

[126] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[127] Rong-Hong Jan and Maw-Sheng Chern. Nonlinear integer bilevel programming. *European Journal of Operational Research*, 72(3):574–587, 1994.

[128] Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.

[129] Napsu Karmitsa, Adil M. Bagirov, and Sona Taheri. New diagonal bundle method for clustering problems in large data sets. *European Journal of Operational Research*, 263(2):367–379, 2017.

[130] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[131] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proceedings of the International Conference on Computer Aided Verification*, pages 97–117, 2017.

[132] Shuichi Kawano, Hironori Fujisawa, Toyoyuki Takada, and Toshihiko Shiroishi. Sparse principal component regression with adaptive loading. *Computational Statistics & Data Analysis*, 89:192–203, 2015.

[133] Carl T Kelley. *Iterative methods for optimization*. Society for Industrial and Applied Mathematics, 1999.

[134] Abolfazl Keshvari. Segmented concave least squares: A nonparametric piecewise linear regression. *European Journal of Operational Research*, 266(2):585–594, 2018.

[135] Elias B. Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 724–731, 2016.

[136] Elias B. Khalil, Bistra Dilkina, George L. Nemhauser, Shabbir Ahmed, and Yufen Shao. Learning to run heuristics in tree search. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 659–666, 2017.

[137] Elias Boutros Khalil, Amrita Gupta, and Bistra Dilkina. Combinatorial attacks on binarized neural networks. Technical report, arXiv preprint 1810.03538, 2018.

[138] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. Technical report, arXiv preprint 1412.6980, 2014.

[139] Ted D Klastorin. The p-median problem for cluster analysis: A comparative test using the mixture model approach. *Management Science*, 31(1):84–95, 1985.

[140] Teresa Klatzer and Thomas Pock. Continuous hyper-parameter learning for support vector machines. In *Proceedings of the Computer Vision Winter Workshop*, pages 39–47, 2015.

[141] Stefan Kramer, Gerhard Widmer, Bernhard Pfahringer, and Michael De Groeve. Prediction of ordinal classes using regression trees. *Fundamenta Informaticae*, 47(1-2):1–13, 2001.

[142] M. Kraus, S. Feuerriegel, and A. Oztekin. Deep learning in business analytics and operations research: Models, applications and managerial implications. Technical report, arXiv preprint 1806.10897, 2018.

[143] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[144] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. Technical report, arXiv preprint 1611.01236, 2016.

[145] Renata Krystyna Kwatera and Bruno Simeone. Clustering heuristics for set covering. *Annals of Operations Research*, 43(5):295–308, 1993.

[146] Gert R. G. Lanckriet, Laurent El Ghaoui, Chiranjib Bhattacharyya, and Michael I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2002.

[147] Yann Lecun, Sumit Chopra, Raia Hadsell, Marc Aurelio Ranzato, and Fu Jie Huang. *A tutorial on energy-based learning*. MIT Press, 2006.

[148] Yann LeCun et al. Generalization and network design strategies. In *Connectionism in perspective*, volume 19. Citeseer, 1989.

[149] Francesco Leofante, Nina Narodytska, Luca Pulina, and Armando Tacchella. Automated verification of neural networks: Advances, challenges and perspectives. Technical report, arXiv preprint 1805.09938, 2018.

[150] Mark Lewis, Haibo Wang, and Gary Kochenberger. Exact solutions to the capacitated clustering problem: A comparison of two models. *Annals of Data Science*, 1(1):15–23, 2014.

[151] Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-Rao metric, geometry, and complexity of neural networks. Technical report, arXiv preprint 1711.01530, 2017.

[152] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. Technical report, arXiv preprint 1312.4400, 2013.

[153] Ji Liu and Xiaojin Zhu. The teaching dimension of linear learners. *The Journal of Machine Learning Research*, 17(1):5631–5655, 2016.

[154] Andrea Lodi and Giulia Zarpellon. On learning and branching: a survey. *TOP*, 25(2):207–236, 2017.

[155] Michele Lombardi, Michela Milano, and Andrea Bartolini. Empirical decision model learning. *Artificial Intelligence*, 244:343–367, 2017.

[156] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the International Conference on Knowledge Discovery in Data Mining*, pages 641–647, 2005.

[157] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[158] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. Technical report, arXiv preprint 1706.06083, 2017.

[159] Feng Mai, Michael J. Fry, and Jeffrey W. Ohlmann. Model-based capacitated clustering with posterior regularization. *European Journal of Operational Research*, 271(2):594–605, 2018.

[160] Sebastián Maldonado, Juan Pérez, Richard Weber, and Martine Labbé. Feature selection for support vector machines via mixed integer linear programming. *Information Sciences*, 279:163–175, 2014.

[161] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2871–2877, 2015.

[162] Alan Miller. *Subset selection in regression*. Chapman and Hall/CRC, 2002.

[163] Velibor V Mišić. Optimization of tree ensembles. Technical report, arXiv preprint 1705.10883, 2017.

[164] Ryuhei Miyashiro and Yuichi Takano. Mixed integer second-order cone programming formulations for variable selection in linear regression. *European Journal of Operational Research*, 247(3):721–731, 2015.

[165] Guido Montúfar. Notes on the number of linear regions of deep neural networks. Technical report, Max Planck Institute for Mathematics in the Sciences, 2017.

[166] Gregory Moore, Charles Bergeron, and Kristin P Bennett. Model selection for primal SVM. *Machine learning*, 85(1-2):175–208, 2011.

[167] Michael J. Mortenson, Neil F. Doherty, and Stewart Robinson. Operational research from taylorism to terabytes: A research agenda for the analytics age. *European Journal of Operational Research*, 241(3):583–595, 2015.

[168] John M Mulvey and Harlan P Crowder. Cluster analysis: An application of Lagrangian relaxation. *Management Science*, 25(4):329–340, 1979.

[169] Marcos Negreiros and Augusto Palhano. The capacitated centred clustering problem. *Computers & operations research*, 33(6):1639–1663, 2006.

[170] Siqi Nie, Cassio P De Campos, and Qiang Ji. Learning bounded tree-width Bayesian networks via sampling. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 387–396. Springer, 2015.

[171] Siqi Nie, Denis D Mauá, Cassio P De Campos, and Qiang Ji. Advances in learning Bayesian networks of bounded treewidth. In *Advances in Neural Information Processing Systems*, pages 2285–2293, 2014.

[172] Siqi Nie, Denis D Maua, Cassio P de Campos, and Qiang Ji. Advances in learning bayesian networks of bounded treewidth. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2285–2293. Curran Associates, Inc., 2014.

[173] Pekka Parviainen, Hossein Shahrabi Farahani, and Jens Lagergren. Learning bounded tree-width Bayesian networks using integer linear programming. In *Artificial Intelligence and Statistics*, pages 751–759, 2014.

[174] Kaustubh R Patil, Jerry Zhu, Ł ukasz Kopeć, and Bradley C Love. Optimal teaching for limited-capacity human learners. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2465–2473. Curran Associates, Inc., 2014.

[175] Harold J. Payne and William S. Meisel. An algorithm for constructing optimal binary decision trees. *IEEE Transactions on Computers*, 26(9):905–916, 1977.

[176] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Alexandre Passos Jake Vanderpla and, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and douard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[177] Selwyn Piramuthu. Evaluating feature selection methods for learning in data mining applications. *European Journal of Operational Research*, 156(2):483–494, 2004.

[178] Tomaso Poggio, Kenji Kawaguchi, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Xavier Boix, Jack Hidary, and Hrushikesh Mhaskar. Theory of deep learning iii: explaining the non-overfitting puzzle. Technical report, arXiv preprint 1801.00173, 2017.

[179] Robert Reris and Jean Paul Brooks. Principal component analysis and optimization: a tutorial. Technical report, Virginia Commonwealth University, 2015.

[180] Herbert Robbins and Sutton Monro. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pages 102–109. Springer, 1985.

[181] Riccardo Rovatti, Claudia D'Ambrosio, Andrea Lodi, and Silvano Martello. Optimistic MILP modeling of non-linear optimization problems. *European Journal of Operational Research*, 239(1):32–45, 2014.

[182] Burcu Sağlam, F. Sibel Salman, Serpil Sayın, and Metin Türkay. A mixed-integer programming approach to the clustering problem with an application in customer segmentation. *European Journal of Operational Research*, 173(3):866–879, 2006.

[183] Éverton Santi, Daniel Aloise, and Simon J. Blanchard. A model for clustering data from heterogeneous dissimilarities. *European Journal of Operational Research*, 253(3):659–672, 2016.

[184] Mauro Scanagatta, Giorgio Corani, Cassio P de Campos, and Marco Zaffalon. Learning treewidth-bounded bayesian networks with thousands of variables. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1462–1470. Curran Associates, Inc., 2016.

[185] Stephan Scheuerer and Rolf Wendolsky. A scatter search heuristic for the capacitated clustering problem. *European Journal of Operational Research*, 169(2):533–547, 2006.

[186] Anita Schöbel. Locating least-distant lines in the plane. *European Journal of Operational Research*, 106(1):152–159, 1998.

[187] Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. Technical report, arXiv preprint 1711.02114, 2017.

[188] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204, 2018.

[189] Amnon Shashua and Anat Levin. Ranking with large margin principle: Two approaches. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 961–968. MIT Press, 2003.

[190] Ayumi Shinohara and Satoru Miyano. Teachability in computational learning. *New Generation Computing*, 8(4):337–347, Feb 1991.

[191] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[192] Raymond J. Solomonoff. An inductive inference machine. In *IRE Convention Record, Section on Information Theory*, volume 2, pages 56–62, 1957.

[193] Heda Song, Isaac Triguero, and Ender Özcan. A review on the self and dual interactions between machine learning and optimisation. *Progress in Artificial Intelligence*, 8(2):143–165, 2019.

[194] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3517–3529. Curran Associates, Inc., 2017.

[195] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. Technical report, arXiv preprint 1312.6199, 2013.

[196] Ryuta Tamura, Ken Kobayashi, Yuichi Takano, Ryuhei Miyashiro, Kazuhide Nakata, and Tomomi Matsui. Best subset selection for eliminating multicollinearity. *Journal of the Operations Research Society of Japan*, 60(3):321–336, 2017.

[197] Ryuta Tamura, Ken Kobayashi, Yuichi Takano, Ryuhei Miyashiro, Kazuhide Nakata, and Tomomi Matsui. Mixed integer quadratic optimization formulations for eliminating multicollinearity based on variance inflation factor. *Journal of Global Optimization*, 73(2):431–446, 2019.

[198] Pakize Taylan, G-W Weber, and Amir Beck. 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, 2007.

[199] Sunil Tiwari, H.M. Wee, and Yosef Daryanto. Big data analytics in supply chain management between 2010 and 2016: Insights to industries. *Computers & Industrial Engineering*, 115:319–330, 2018.

[200] Vincent Tjeng and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. Technical report, arXiv preprint 1711.07356, 2017.

[201] Alejandro Toriello and Juan Pablo Vielma. Fitting piecewise linear continuous functions. *European Journal of Operational Research*, 219(1):86–95, 2012.

[202] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. Technical report, arXiv preprint 1705.07204, 2017.

[203] Vladimir Vapnik. *Statistical learning theory*, volume 3. Wiley, New York, 1998.

[204] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[205] Sicco Verwer and Yingqian Zhang. 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*, pages 94–103, 2017.

[206] Sicco Verwer, Yingqian Zhang, and Qing Chuan Ye. Auction optimization using regression trees and linear models as integer programs. *Artificial Intelligence*, 244:368–395, 2017.

[207] Juan Pablo Vielma, Shabbir Ahmed, and George Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research*, 58(2):303–315, 2010.

[208] Roman Vclavk, Antonn Novk, Pemysl cha, and Zdenk Hanzlek. Accelerating the branch-and-price algorithm using machine learning. *European Journal of Operational Research*, 271(3):1055–1069, 2018.

[209] Gang Wang, Angappa Gunasekaran, Eric W.T. Ngai, and Thanos Papadopoulos. Big data analytics in logistics and supply chain management: Certain investigations for research and applications. *International Journal of Production Economics*, 176:98–110, 2016.

[210] Hua Wang, Chris Ding, and Heng Huang. Multi-label linear discriminant analysis. In *Proceedings of the European Conference on Computer Vision*, pages 126–139, 2010.

[211] Li Wang, Ji Zhu, and Hui Zou. The doubly regularized support vector machine. *Statistica Sinica*, 16(2):589, 2006.

[212] Yizhen Wang and Kamalika Chaudhuri. Data poisoning attacks against online learning. Technical report, arXiv preprint 1808.08994, 2018.

[213] Yuan Wang, Dongxiang Zhang, Ying Liu, Bo Dai, and Loo Hay Lee. Enhancing transportation systems via deep learning: A survey. *Transportation Research Part C: Emerging Technologies*, 99:144–163, 2019.

[214] Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. A survey on neural architecture search. Technical report, arXiv preprint arXiv:1905.01392, 2019.

[215] Stephen J. Wright. Optimization algorithms for data analysis. Technical report, Optimization Online, 2016.

[216] Changhe Yuan and Brandon Malone. Learning optimal Bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, 48:23–65, 2013.

[217] Jerry Zhu. Machine teaching for bayesian learners in the exponential family. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1905–1913. Curran Associates, Inc., 2013.

[218] Ji Zhu, Saharon Rosset, Robert Tibshirani, and Trevor J. Hastie. 1-Norm support vector machines. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 49–56. MIT Press, 2004.

[219] Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4083–4087, 2015.

[220] Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N Rafferty. An overview of machine teaching. Technical report, arXiv preprint 1801.05927, 2018.

[221] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the International Conference on Machine Learning*, pages 928–936, 2003.

[222] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.