Optimal representative sample weighting

Shane Barratt[1] · **Guillermo Angeris[1]** · **Stephen Boyd[1]**

Abstract

We consider the problem of assigning weights to a set of samples or data records, with the goal of achieving a representative weighting, which happens when certain sample averages of the data are close to prescribed values. We frame the problem of finding representative sample weights as an optimization problem, which in many cases is convex and can be efficiently solved. Our formulation includes as a special case the selection of a fixed number of the samples, with equal weights, i.e., the problem of selecting a smaller representative subset of the samples. While this problem is combinatorial and not convex, heuristic methods based on convex optimization seem to perform very well. We describe our open-source implementation `rsw` and apply it to a skewed sample of the CDC BRFSS dataset.

1 Introduction

We consider a setting where we have a set of data samples that were not uniformly sampled from a population, or where they were sampled from a different population than the one from which we wish to draw some conclusions. A common approach is to assign *weights* to the samples, so the resulting weighted distribution is *representative* of the population we wish to study. Here, representative means that with the weights, certain expected values or probabilities match or are close to known values for the population we wish to study.

As a very simple example, consider a dataset where each sample is associated with a person. Our dataset is 70% female, whereas we would like to draw conclusions about a population that is 50% female. A simple solution is to down-weight the female samples and up-weight the male samples in our dataset, so the weighted fraction of females is 50%. As a more sophisticated example, suppose we have multiple groups, for example, various combinations of sex, age group, income level, and education, and our goal is to find weights

for our samples, so the fractions of all these groups match or approximate known fractions in the population we wish to study. In this case, there will be many possible assignments of weights that match the given fractions, and we need to choose a reasonable one. One approach is to maximize the entropy of the weights, subject to matching the given fractions. This problem generally does not have an analytical solution, but it can be solved by various methods, for example, *raking* or *iterative proportional fitting* (Lumley 2011, Sect. 7), or the methods described in this paper.

We formulate a general method for choosing weights that balances two objectives: making the weighted sample representative, i.e., matching some known fractions or other statistics, and having weights with desirable properties, such as being close to uniform. We pose this as an optimization problem. In many practical cases, the problem is convex and so can be solved efficiently (Boyd and Vandenberghe 2004); in others, the problem can be expressed as a mixed-integer convex optimization problem, which in general is hard to solve exactly, but for which simple and effective heuristics can be developed.

Our method allows for a variety of measures of how representative a weighted dataset is, and a variety of other desirable properties for the weights themselves. Simple examples of measures of representativeness include allowing the user to specify ranges for fractions or expected values, instead of insisting that the values are matched precisely. For example, we might specify that the weighted sample should have a

✉ Shane Barratt
  sbarratt@stanford.edu

  Guillermo Angeris
  angeris@stanford.edu

  Stephen Boyd
  boyd@stanford.edu

[1] Electrical Engineering Department, Stanford University, Stanford, USA

fraction of females between 49% and 51%, instead of insisting that it should be exactly 50%. As a simple example of desirable properties for the weights, we can specify minimum and maximum allowable values for the weights, e.g., we can require that the weights range between one half and twice the uniform weighting, so no sample is down- or up-weighted by more than a factor of two.

One particularly interesting constraint we can impose on the weights is that they are all zero, except for a specified number k of them, and those must have the same weight, $1/k$. The weighted data set is then nothing more than a subset of k of our original samples, with the usual uniform weights. This means that we are selecting a set of k samples from our original dataset that is representative. This can be useful when we need to carry out some expensive further testing of some of the samples, and we wish to choose a subset of them that is representative.

In this paper, we present a general framework for choosing weights and a general algorithm for solving the associated optimization problems. Our method, which is based on an operator splitting method called ADMM (alternating directions method of multipliers), solves the problem exactly when it is convex, and is a good heuristic when the problem is not convex. We have implemented the method as an open-source software package in Python, called `rsw` (for representative sample weights). It handles a wide variety of weight selection problems, and scales to very large problems with hundreds of thousands or millions of samples, making it suitable for almost all practical surveys.

Related work Survey weighting is a well-studied problem in statistics (see, e.g., Valliant et al. (2013) for an overview). The simplest technique for survey weighting is post-stratification (Neyman 1934). When the samples can be divided into a finite number of groups, and we want the fraction of samples in each group to be equal to some given fraction, post-stratification adjusts the weights of each group using a simple formula so that the fraction of samples in the weighted groups match the given fractions (Holt and Smith 1979). For example, if we had 4 female samples and 6 male samples, and we wanted half to be female, we would give each female sample a weight of 0.125 and each male sample a weight of 0.083. Post-stratification, when applied to multiple attributes, requires the groups to be constructed as each combination of the attributes, which might lead to many groups and be undesirable (Lumley 2011, Sect. 7.3).

An alternative to post-stratification, when there are multiple attributes, is to match the marginal distribution for each attribute instead of the full joint distribution, while maximizing the entropy of the sample weights. This problem is exactly the maximum entropy problem described in Sect. 3.1. The most common iterative algorithm used in practice is raking (also known as iterative proportional fitting or rim weighting) (Yule 1912; Kruithof 1937; Deming and Stephan 1940),

which works by cycling through each attribute and performing post-stratification in order to gradually update the sample weights. Raking is the standard weighting method used by many public pollsters, according to the Pew Research Center (Mercer et al. 2018). It was observed that a version of raking can be seen as coordinate ascent on the dual of the maximum entropy problem, which is guaranteed to converge (Deville et al. 1993; Teh and Welling 2003) (see Appendix A). Other optimization-based formulations of survey weighting can be found in Fu et al. (2019), She and Tang (2019).

There are a number of generalizations to the raking procedure, that are each for different regularizers (Deville et al. 1993, Sect. 3). Some of these are linear weighting (Bethlehem and Keller 1987) (which uses the Pearson $\mathcal{X}^2$-divergence between the weights and the prior on the weights as the regularizer), logit weighting, and truncated linear weighting. Each of these has a corresponding generalized raking procedure, yet all of these generalizations are instances of our formulation, and can be easily solved using our ADMM algorithm even when the loss or regularizer is nonsmooth. Another technique is logistic regression weighting, which regresses the probability of sampling each sample on the sample attributes, and uses the reciprocal of the predicted sampling probabilities as weights (Lepkowski et al. 1989; Iannacchione et al. 1991). We note that various heuristic techniques have been proposed for constraining the weights in the raking algorithm (Kalton and Flores-Cervantes 2003, Sect. 3), but, in our formulation, these can be easily expressed as constraints included in the regularizer. To our best knowledge, the representative selection problem, described in Sect. 3.2, has not appeared in the literature. The closest problem is representative subset selection (Daszykowski et al. 2002), which tries to select a set of samples that are as different as possible from one another.

Once sample weights have been found, they can be used in a Horvitz–Thompson estimator to estimate various quantities (Horvitz and Thompson 1952. They can also be used in downstream statistical analysis, e.g., linear regression (Lavallée and Beaumont 2015). We discuss these and some other applications for weights in Sect. 4. For a discussion of whether weighting is appropriate or not for a particular dataset or goal, see, e.g., Kish (1992). (Our focus is on finding weights, and not whether or not using them is appropriate.)

Another important aspect of survey weighting is nonresponse weighting (also known as propensity weighting or direct standardization). Nonresponse weighting is the first part of the weighting process, which adjusts the weight of each sample based on the probability that a sample with those attributes would not respond to the survey (Lumley 2011, Sect. 9). The most common technique is to construct a logistic regression model, which regresses the response probability on the sample attributes (Heckman 1976). We assume

that nonresponse weighting has already been performed or that there is no nonresponse bias in the dataset.

Outline In Sect. 2, we describe the setting of sample weighting and the optimization problem of finding representative sample weights. In Sect. 3, we provide a number of examples of the representative sample weighting problem. In Sect. 5, we give a solution method based on ADMM. In Sect. 4, we describe a number of applications once sample weights have been computed. In Sect. 6, we demonstrate our method on annual survey data from the US CDC. We conclude in Sect. 7 with extensions and variations.

Appendix A gives a proof that raking is equivalent to performing block coordinate ascent on the dual variables for a particular instance of the representative sample weighting problem, and Appendix B gives a table of desired expectations for the provided examples.

2 Representative sample weighting

In this section, we describe the general problem of representative sample weighting.

Samples and weights We are given n samples or data records $x_1, \ldots, x_n \in \mathcal{X}$. For our purposes, the feature values x_i and the feature set $\mathcal{X}$ do not matter; they will enter our problem in a very specific way, described below. We will assign a weight $w_i \in \mathbf{R}_+$ to each sample x_i, $i = 1, \ldots, n$, where $\mathbf{1}^T w = 1$ and $\mathbf{1}$ is the all-ones vector. The sample weights induce a distribution on $\{x_1, \ldots, x_n\}$, with $\mathbf{Prob}(x_i) = w_i$, $i = 1, \ldots, n$. Our goal is to choose these sample weights.

Expected values of functions We are given real-valued functions $F_j : \mathcal{X} \to \mathbf{R}$, $j = 1, \ldots, m$. We let $f \in \mathbf{R}^m$ denote the expected values of these functions under the induced distribution,

$$f_j = \sum_{i=1}^{n} w_i F_j(x_i), \quad j = 1, \ldots, m.$$

We express this as $f = Fw$, where $F \in \mathbf{R}^{m \times n}$ is the matrix with entries $F_{ij} = F_i(x_j)$. In the sequel, we will only need this matrix F; we will not otherwise need the particular values of x_i or the functions F_j.

Probabilities We note that when the function F_j is the $\{0, 1\}$ indicator function of a set $S \subseteq \mathcal{X}$, f_j is the probability of S under the induced distribution. As an elementary example of this, S could be the set of examples that are female (which is encoded in x), in which case f_j is the probability, under the weights w, that the sample is female.

As an extension of this, suppose $F_1, \ldots, F_m$ are the $\{0, 1\}$ indicator functions of the sets $S_1, \ldots, S_m \subseteq \mathcal{X}$, and each example is in exactly one of these sets (i.e., they are a partition). In this case $f = Fw$ is a distribution on $\{1, \ldots, m\}$, i.e., $f \geq 0$ (elementwise) and $\mathbf{1}^T f = 1$. As an elementary example, these could be a partition of samples into female and male, and a number of age groups; f_j is then the probability under the weighted distribution that an example is in group j, which is some combination of sex and age group.

Desired expected values and loss function Our task is to choose the weights $w \in \mathbf{R}_+^n$ so that this induced distribution is *representative*, which means that the expected values of the functions of x_i are near some target or desired values, denoted $f^{\text{des}} \in \mathbf{R}^m$. To get a numerical measure of how representative the induced distribution is, we introduce a loss function $\ell : \mathbf{R}^m \times \mathbf{R}^m \to \mathbf{R} \cup \{+\infty\}$, where $\ell(f, f^{\text{des}})$ measures our dissatisfaction with a particular induced expected value f and desired expected value f^{des}. We use infinite values of ℓ to denote constraints on f and f^{des}. For example, to constrain $f = f^{\text{des}}$, we let $\ell(f, f^{\text{des}}) = +\infty$ for $f \neq f^{\text{des}}$, and $\ell(f, f^{\text{des}}) = 0$ for $f = f^{\text{des}}$. Other common losses include the sum squares loss $\| f - f^{\text{des}} \|_2^2$, or the ℓ_1 loss $\| f - f^{\text{des}} \|_1$.

When f is a distribution on $\{1, \ldots, m\}$ (i.e., for all w with $w \geq 0$, $\mathbf{1}^T w = 1$, we have $Fw \geq 0$ and $\mathbf{1}^T Fw = 1$), a reasonable loss is the Kullback–Leibler (KL) divergence (Kullback and Leibler 1951) from f^{des}, a desired target distribution,

$$\ell(f, f^{\text{des}}) = \sum_{i=1}^{m} f_i \log(f_i / f_i^{\text{des}}).$$

Here we assume all entries of f together form a distribution; if there are multiple groups of entries in f that form distributions, we can use the sum of the KL divergences from their associated target distributions. As an example and special case suppose f_i is the probability that $x \in S_i$, where $S_i \subseteq \mathcal{X}$ are sets (not necessarily a partition), for which we have target values f_i^{des}. Then, $1 - f_i$ is the probability that $x \notin S_i$. A reasonable loss is the sum of the KL divergence from these two probabilities and their associated desired probabilities, i.e.,

$$\ell(f, f^{\text{des}}) = \sum_{i=1}^{m} \left(f_i \log \left(\frac{f_i}{f_i^{\text{des}}} \right) + (1 - f_i) \log \left(\frac{1 - f_i}{1 - f_i^{\text{des}}} \right) \right).$$

Some other reasonable losses when f is a distribution on $\{1, \ldots, m\}$ are total variation (otherwise known as the ℓ_1-norm) and earth mover's distance (Peyré and Cuturi 2019).

Regularizer We also have preferences and requirements on the choice of weights, beyond $w \geq 0$ and $\mathbf{1}^T w = 1$, unrelated to the data samples and functions. We express these using a regularization function $r : \mathbf{R}^n \to \mathbf{R} \cup \{\infty\}$, where we prefer smaller values of $r(w)$. Again, we use infinite values of r to denote constraints on w. For example, to constrain $w \geq w^{\text{min}}$ for $w^{\text{min}} \in \mathbf{R}_+^n$, we let $r(w) = +\infty$ whenever $w < w^{\text{min}}$.

An important example of a regularizer is the negative entropy, with

$$r(w) = \sum_{i=1}^{n} w_i \log w_i,$$

which is the same as the KL divergence from the distribution given by w and the uniform distribution $\mathbf{1}/n$ on $\{1, \ldots, n\}$. This regularizer expresses our desire that the weights have high entropy, or small KL deviation from uniform. A simple extension of this regularizer is the KL deviation from a (nonuniform) target distribution w^{tar}.

Regularizers can be combined, as in

$$r(w) = \begin{cases} \sum_{i=1}^{n} w_i \log w_i & (1/(\kappa n))\mathbf{1} \le w \le (\kappa/n)\mathbf{1} \\ \infty & \text{otherwise,} \end{cases} \quad (1)$$

where $\kappa > 1$ is a given hyper-parameter. This regularizer combines negative entropy with the constraint that no sample is up- or down-weighted by a factor more than κ.

Representative sample weighting problem We propose to choose weights as a solution of the *representative sample weighting* problem

$$\begin{aligned} \text{minimize} \quad & \ell(f, f^{\text{des}}) + \lambda r(w) \\ \text{subject to} \quad & f = Fw, \quad w \ge 0, \quad \mathbf{1}^T w = 1, \end{aligned} \quad (2)$$

with variables w and f, and positive hyper-parameter λ. The objective is a weighted sum of the loss, which measures how unrepresentative the induced distribution is, and the regularization, which measures our displeasure with the set of weights. The hyper-parameter λ is used to control the trade-off between these two objectives.

The representative sample weighting problem (2) is specified by the loss function ℓ and desired expected values f^{des}, the regularizer r, and the matrix F which gives the function values $F_i(x_j)$.

Convexity When ℓ is convex in its first argument and r is convex, problem (2) is a convex optimization problem and so can be efficiently solved (Boyd and Vandenberghe 2004). Many interesting and useful instances of the representative sample weighting problem are convex; a few are not. The cases when the problem is not convex can be solved using global optimization methods; we will also recommend some simple but effective heuristics for solving such problems approximately.

3 Examples

In this section, we give two examples of the representative sample weighting problem.

3.1 Maximum entropy weighting

The first case we consider is where we want to find the maximum entropy sample weights that exactly match the desired

expected values. This is a representative sample weighting problem, with the loss and regularizer

$$\ell(f, f^{\text{des}}) = \begin{cases} 0 & f = f^{\text{des}}, \\ +\infty & \text{otherwise,} \end{cases} \quad r(w) = \sum_{i=1}^{n} w_i \log w_i.$$

Since both the loss and regularizer are convex, problem (2) is a convex optimization problem. We call the (unique) solution to problem (2) with this loss and regularizer the *maximum entropy weighting*. The maximum entropy weighting is the most random or evenly spread out weighting that exactly matches the desired expected values.

The maximum entropy weighting problem is convex and readily solved by many methods. The typical complexity of such methods is order nm^2, i.e., linear in the number of samples n and quadratic in the number of functions m (see, e.g., Boyd and Vandenberghe (2004), Sect. 11.5). This complexity assumes that the matrix F is treated as dense; by exploiting sparsity in F, the complexity of solving the problem can be lower.

Simple example As a very simple example, suppose $m = 1$ and F_1 is the $\{0, 1\}$ indicator function of the sample being female, so $f = Fw$ is the probability of being female under the weighted distribution, and we take $f^{\text{des}} = 0.5$, which means we seek weights for which the probability of female is 0.5. (This problem has an elementary and obvious solution, where we weight all female samples the same way, and all nonfemale samples the same way. But our interest is in more complex problems that do not have simple analytical solutions.)

Variations We have already mentioned several variations, such as adding bounds on w, or replacing the negative entropy with the KL divergence from a target weighting. As another extension, we can modify the loss to allow the expected function values f to be close to, but not exactly, their desired values. As a simple version of this, we can require that $f^{\text{min}} \le f \le f^{\text{max}}$, where $f^{\text{min}}, f^{\text{max}} \in \mathbf{R}^m$ are given lower and upper limits (presumably with $f^{\text{min}} \le f^{\text{des}} \le f^{\text{max}}$). This corresponds to the loss function

$$\ell(f, f^{\text{des}}) = \begin{cases} 0 & f^{\text{min}} \le f \le f^{\text{max}}, \\ +\infty & \text{otherwise,} \end{cases}$$

the indicator function of the constraint $f^{\text{min}} \le f \le f^{\text{max}}$.

In our simple example above, for example, we might take $f^{\text{min}} = 0.45$ and $f^{\text{max}} = 0.55$. This means that we will accept any weights for which the probability of female is between 0.45 and 0.55. (This problem also has an elementary analytical solution.)

3.2 Representative selection

In the second example, we consider the problem of selecting $k < n$ samples, with equal weights, that are as representative as possible. Thus, our regularizer has the form

$$r(w) = \begin{cases} 0 & w \in \{0, 1/k\}^n, \\ +\infty & \text{otherwise.} \end{cases} \tag{3}$$

This regularizer is infinite unless we have $w_i = 1/k$ for $i \in \mathcal{I}$, and $w_i = 0$ for $i \notin \mathcal{I}$, where $\mathcal{I} \subset \{1, \ldots, n\}$ and $|\mathcal{I}| = k$. In this case, the induced distribution is uniform on x_i for $i \in \mathcal{I}$.

With this constraint, it is unlikely that we can achieve $f = f^{\text{des}}$ exactly, so we use a loss function such as $\ell(f, f^{\text{des}}) = \|f - f^{\text{des}}\|_2^2, \ell(f, f^{\text{des}}) = \|f - f^{\text{des}}\|_1$, or the KL divergence to f^{des} if f is a distribution.

We will refer to this problem as *representative selection*. Roughly speaking, the goal is to choose a subset of size k from the n original data points, such that this subset is as representative as possible. The problem is interesting even when we take $f^{\text{des}} = (1/n)F\mathbf{1}$, in which case the problem is to choose a subset of the original samples, of size k, that approximates the means of the functions on the original dataset.

Representative selection is a combinatorial optimization problem of selecting k out of n original data points, of which there are $\binom{n}{k}$ choices. This problem can be difficult to solve exactly, since you can solve the 0–1 integer problem by solving n instances of the representative selection problem (Karp 1972). Representative selection can be reformulated as a mixed-integer convex program, for which there exist modern solvers (e.g., ECOS Domahidi et al. 2013, GUROBI (Gurobi Optimization 2020), and MOSEK ApS 2020, Sect. 9) that can often find global solutions of small problems in a reasonable amount of time. For the representative selection problem, we describe an ADMM algorithm in Sect. 5 that can quickly find an approximate solution, which appears to have good practical performance and can be used when n is very large. We also note that the representative selection problem is similar in nature to the sparse approximation problem, which can be approximately solved by heuristics such as basis or matching pursuit (Chen et al. 2001).

4 Applications

In this section, we describe what to do once sample weights have been computed.

Direct use of weights The most common use of sample weights is to use them as weights in each subsequent step of a traditional machine learning or statistical analysis pipeline. Most machine learning and statistical methods are able to incorporate sample weights, by in some way modulating the importance of each sample by its weight in the fitting process. As a result, samples with higher weights affect the ultimate model parameters more than those with lower weights. As a simple example, suppose we have another function $g : \mathcal{X} \rightarrow \mathbf{R}$ for which we want to compute its mean. Our naïve estimate of the mean would be the sample average, $\sum_{i=1}^n (1/n) g(x_i)$; using the sample weights, our estimate of the mean is $\sum_{i=1}^n w_i g(x_i)$. (This is the aforementioned Horvitz–Thompson estimator.)

As another example, suppose we have additional features of each sample denoted $a_1, \ldots, a_n \in \mathbf{R}^p$ and outcomes $b_1, \ldots, b_n \in \mathbf{R}$ and we wish to fit a linear regression model, i.e., $\theta^T a_i \approx b_i$ for some $\theta \in \mathbf{R}^p$. With sample weights, the fitting problem becomes

$$\text{minimize} \quad \sum_{i=1}^n w_i (\theta^T a_i - b_i)^2.$$

Re-sampling Another option is to use the sample weights to re-sample the data and then use the new set of samples for further analysis (Yu 2002; Estabrooks et al. 2004). That is, we create a new set of N samples $\tilde{x}_1, \ldots, \tilde{x}_N$, using the distribution

$$\mathbf{Prob}(\tilde{x}_i = x_i) = w_i, \quad i = 1, \ldots, N.$$

For many machine learning and statistical analysis methods, when N becomes very large, re-sampling the data converges to the approach above of using the weights directly. However, re-sampling can be useful when the machine learning or statistical analysis method cannot incorporate weights.

Selection The representative selection problem, described in Sect. 3.2, can be used to select a subset of the samples that are themselves representative. Representative selection can be useful when we want to select a few of the samples to gather more data on, but it is infeasible (or difficult, or expensive) to gather more data on all of the samples. This can be useful in a two-stage sampling or study, where one first gathers basic (low-cost) data on a large number of samples, and then selects a representative subset of those samples to include in the second (high-cost) study (Lumley 2011, Sect. 8).

For future reference, we mention that a simple heuristic for representative selection comes directly from weighted sampling, described above. We start by solving the maximum entropy weight selection problem and then draw k samples from the associated weighted distribution. This simple scheme can be used as an initialization for a heuristic method for the problem.

5 Solution method

The problem (2) can be solved by many methods, depending on the properties of the loss and regularizer functions.

For example, in the maximum entropy problem, with equality constraints on f or with sum square loss, Newton's method can be effectively used (Boyd and Vandenberghe 2004, Sect. 11). More complex problems, with nondifferentiable loss and regularizer, can be solved using, for example, interior-point methods, after appropriate transformations (Boyd and Vandenberghe 2004, Sect. 11.6). It is very easy to express general convex representative sample weighting problems using domain-specific languages (DSLs) for convex optimization, such as CVX (Grant and Boyd 2008, 2014), YALMIP (Lofberg 2004), CVXPY (Diamond and Boyd 2016; Agrawal et al. 2018), Convex.jl (Udell et al. 2014), and CVXR (Fu et al. 2019). Many of these packages are also able to handle mixed-integer convex programs, e.g., the representative selection problem can be expressed in them. These DSLs canonicalize the (human-readable) problem into a convex or mixed-integer convex cone program and call a numerical solver, such as OSQP (Stellato et al. 2020b), SCS (O'Donoghue et al. 2016), ECOS (Domahidi et al. 2013), MOSEK (Domahidi et al. 2013), or GUROBI (Gurobi Optimization 2020). However, as we will see in Sect. 6, these generic solvers can often be quite slow, taking two orders of magnitude more time to solve the problem compared to the algorithm we describe below. (The advantage of these DSLs is that representative sampling problems can be specified in just a few lines of code.)

In this section, we describe a simple and efficient algorithm for solving problems of the form (2), including losses and regularizers that take on infinite values and are nondifferentiable or nonconvex. While it can at times be slower to achieve comparable accuracy to that of an interior-point method, high accuracy is not needed in practice in the representative sample weighting problem. Our algorithm uses the alternating direction method of multipliers [ADMM; see Boyd et al. (2010)]. When the loss and regularizer are convex, and the problem is feasible, this algorithm is guaranteed to converge to a globally optimal solution. In the case when the loss or regularizer is not convex, our method is a heuristic that will find an approximate solution; it has been observed that these approximate solutions are often good enough for practical purposes and that the approximate solutions can be refined or polished using greedy methods (Diamond et al. 2018).

Problem rewriting In order to apply ADMM to problem (2), we introduce two new redundant variables, resulting in the problem

$$
\begin{array}{ll}
\text{minimize} & \ell(f, f^{\text{des}}) + \lambda r(\tilde{w}) \\
\text{subject to} & f = Fw \\
& \tilde{w} = w = \bar{w} \\
& \bar{w} \geq 0, \quad \mathbf{1}^T \bar{w} = 1,
\end{array}
\tag{4}
$$

with variables $w, \tilde{w}, \bar{w} \in \mathbf{R}^n$ and $f \in \mathbf{R}^m$. The problem data are still the data matrix $F \in \mathbf{R}^{m \times n}$ and the desired values

$f^{\text{des}} \in \mathbf{R}^m$. Problem (4) is said to be in *graph form*, and many algorithms exist that can solve such problems, e.g., ADMM (Parikh and Boyd 2014a) and POGS (Fougner and Boyd 2018).

In (4), we replicate the variable w into three versions, which must all be equal. Each of these versions of w handles one aspect of the problem: $\bar{w}$ is constrained to be a probability distribution; $\tilde{w}$ appears in the regularizer, and w appears in the loss. The technique of replicating a variable and insisting that the different versions be consistent is sometimes called *consensus optimization* (Boyd et al. 2010, Sect. 7.1). By itself, this consensus form is useless. It becomes useful when combined with an operator splitting method that handles the different versions of w separately. We now describe one such operator splitting method, ADMM.

ADMM steps Using the splitting $(f, \tilde{w}, \bar{w})$ and w, the corresponding ADMM steps (see, e.g., Parikh and Boyd (2014b), Sect. 4.4) are

$$
\begin{aligned}
f^{k+1} &= \mathbf{prox}_{(1/\rho)\ell(\cdot, f^{\text{des}})} \left(Fw^k - y^k \right), \\
\tilde{w}^{k+1} &= \mathbf{prox}_{(\lambda/\rho)r} \left(w^k + z^k \right), \\
\bar{w}^{k+1} &= \Pi \left(w^k + u^k \right), \\
w^{k+1} &= \operatorname{argmin}_w \left\| \begin{bmatrix} f^{k+1} - Fw + y^k \\ \tilde{w}^{k+1} - w + z^k \\ \bar{w}^{k+1} - w + u^k \end{bmatrix} \right\|_2^2, \\
y^{k+1} &= y^k + f^{k+1} - Fw^{k+1}, \\
z^{k+1} &= z^k + w^{k+1} - \tilde{w}^{k+1}, \\
u^{k+1} &= u^k + w^{k+1} - \bar{w}^{k+1},
\end{aligned}
\tag{5}
$$

where $y \in \mathbf{R}^m$ and $z, u \in \mathbf{R}^n$ are the (scaled) dual variables for the constraints, and $\rho > 0$ is a given penalty parameter. Here, $\mathbf{prox}_g$ is the proximal operator of the function g (see Parikh and Boyd 2014b, Sect. 1.1) and often has closed-form solutions for many convex (and some nonconvex) functions that appear in practice. The function Π is the projection onto the probability simplex $\{ w \in \mathbf{R}^n \mid w \geq 0, \ \mathbf{1}^T w = 1 \}$.

Convergence When the problem is convex and feasible, the ADMM steps are guaranteed to converge to a global solution for any penalty parameter $\rho > 0$. When the problem is not convex, the ADMM steps are not guaranteed to converge to a global solution, or even converge. However, it has been observed that ADMM is often effective at finding approximate solutions to nonconvex problems (Diamond et al. 2018; Angeris et al. 2019).

Efficient computation of steps We note that the first three steps in (5), which update f, $\tilde{w}$, and $\bar{w}$, are independent and can be parallelized, i.e., carried out simultaneously. Moreover, when ℓ or r is separable or block separable, the proximal operators

can be evaluated on each block independently, leading to additional parallelization.

The update for w^{k+1} can be expressed as the solution to the linear system

$$\begin{bmatrix} 2I & F^T \\ F & -I \end{bmatrix} \begin{bmatrix} w^{k+1} \\ \eta \end{bmatrix} = \begin{bmatrix} q \\ 0 \end{bmatrix},$$

where $\eta = Fw^{k+1} \in \mathbf{R}^m$ is an auxiliary variable and

$$q = F^T(f^{k+1} + y^k) + \tilde{w}^{k+1} + z^k + \bar{w}^{k+1} + u^k,$$

The coefficient matrix in this system of linear equations is quasi-definite (Vanderbei 1995), so it always has an LDL^T factorization with diagonal D. We solve this linear system by performing a sparse LDL^T factorization and caching the factorization between iterations. As a result, after the initial factorization, the cost of each subsequent iteration is that of a back-solve step, which can be much smaller than the initial factorization (Boyd and Vandenberghe 2004, App. C).

There is a large literature on, and many existing libraries for, efficiently computing proximal operators, see, e.g., Parikh and Boyd (2014a, 2013); Stella et al. (2020). We describe here two proximal operators that arise in solving the representative sample weighting problem and are less well known.

Proximal operator for KL divergence The proximal operator for KL divergence can arise both in the loss and regularizer functions. The proximal operator of $r(w) = \sum_{i=1}^n w_i \log(w_i/u_i)$ is given by

$$\mathbf{prox}_{\lambda r}(v)_i = \lambda W\left(u_i e^{v_i/\lambda - 1}/\lambda\right),$$

where W is the Lambert-W or product log function (Lambert 1758).

Proximal operator for representative selection In the case where the regularizer r is nonconvex, its proximal operator is generally difficult to find in practice. However, one special case where it is in fact easy to evaluate is in the representative selection problem (3). In this case, the proximal operator of r is just the projection of v onto the (nonconvex) set

$$\{w \in \{0, 1/k\}^n \mid \mathbf{1}^T w = 1\}.$$

This is easily computed by setting the largest k entries of v to $1/k$ (with ties broken arbitrarily) and the rest to zero (Diamond et al. 2018, Sect. 4.2).

5.1 Implementation

We have implemented the ADMM algorithm described above in an easy-to-use Python package `rsw`, which is freely avail-

able online at

```
www.github.com/cvxgrp/rsw|.
```

The current implementation exploits very little of the parallelization possible; a future version will exploit much more.

The package exposes a single method, `rsw`, which has the signature

```
def rsw(df, funs, losses, reg):
```

It performs representative sample weighting on the pandas DataFrame `df`, using the functions in the list `funs`. (To use `df` directly as F, one would pass in `funs=None`.) We provide a number of common losses and regularizers, and their associated proximal operators. The argument `losses` specifies the losses, applied in block-separable form. One can also specify the scale of each loss, e.g., for $((1/2)(f_1 - 1))^2$, one would pass in the loss `rsw.LeastSquaresLoss(1, scale=.5)`. The argument `reg` specifies the regularizer. You can also specify limits on the entries of w as in (1), e.g.,

```
reg = rsw.EntropyRegularizer(limit = 2)
```

constrains w to not be up- or down-weighted from uniform by a factor of 2, or $1/(2n)\mathbf{1} \le w \le 2/n\mathbf{1}$. At this point, the software package does not support making the loss or regularizer a sum of multiple losses or regularizers. We note, however, that this can easily be accommodated by adding additional consensus variables, and we plan to implement this in a future version of the software.

Packages We use the Python packages `numpy` (Walt et al. 2011) for dense linear algebra, `scipy` (Jones et al. 2001) for sparse linear algebra, `pandas` (McKinney 2010) for data manipulation, and `qdldl` (Stellato et al. 2020b, a) for the sparse quasi-definite LDL^T factorization.

6 Numerical experiments

We present here an application of the methods described above to a large dataset from the US Center for Disease Control (CDC). We chose examples that are a bit more complex that might be needed, to demonstrate the method's flexibility and scalability. We note that, in practice, the functions and losses used would likely be much simpler, although they need not be. All experiments were conducted on a single core of an Intel i7-8700K CPU with 64GB of memory, which, at the time of writing, was a standard consumer desktop CPU.

CDC BRFSS dataset Each year, the US Center for Disease Control (CDC) conducts health-related telephone surveys of hundreds of thousands of US citizens. (This is the world's largest telephone survey.) The data are collected as part of

the Behavioral Risk Factor Surveillance System (BRFSS) and are freely available online (Center for Disease Control and Prevention (CDC) 2018a). We consider the 2018 version of this dataset, which contains 437436 responses to 275 survey questions. The data include a variety of demographic and health-related attributes (Center for Disease Control and Prevention (CDC) 2018b). We consider the following columns in the dataset:

– *State or territory*. There are 53 possible values, e.g., NM (New Mexico) or PR (Puerto Rico).
– *Age group*. The 6 possible values are 18–24, 25–34, 35–44, 45–54, 55–65, and 65+.
– *Sex*. The 2 possible values are male and female.
– *Education level*. The 4 possible values are elementary, high school, some college, and college.
– *Income*. The 8 possible values are 0–10k, 10–15k, 15–20k, 20–25k, 25–35k, 35–50k, 50–75k, and 75k+.
– *Reported general health*. The 5 possible values are excellent, very good, good, fair, and poor.

Functions As our first group of functions, we consider the {0, 1} indicator functions of each possibility of state and age group, of which there are 318. (The expected values of these functions are the probabilities of each state/age group). Our next two functions are the indicator function of the sex being male or female. Our next group of functions are the indicator functions of each possibility of education level and income, of which there are 32. The final group of functions are the indicator functions of each possibility of reported general health, of which there are 5. In total, we have $m = 357$ functions. The resulting matrix F has over 150 million entries and around 2% of its values are missing. It is sparse, with a density of 3.5%.

Our choice of these particular 357 functions is relatively arbitrary (although they are not unreasonable).

Desired expected values In all experiments, we take the averages across the entire dataset as the desired expected values, or

$$f^{\text{des}} = (1/n)F\mathbf{1}.$$

We ignore missing entries in the computation. The desired expected values are given in tables that can be found in Appendix B.

Constructing a skewed subsample We generate a smaller dataset composed of 10000 samples by generating a skewed sample from the dataset. Denote the ith column of F by $x_i \in \mathbf{R}^m$. As described above, x_i contains a one-hot encoding of various things (state and age, sex, education and income, and health). We generate a skewed sampling distribution from the dataset by first sampling $c \sim \mathcal{N}(0, (1/4)I)$, and then letting

$$\pi_i = \exp(c^T x_i) \bigg/ \sum_{j=1}^{n} \exp(c^T x_j), \quad i = 1, \dots, n.$$

(By construction, $\pi \geq 0$ and $\mathbf{1}^T \pi = 1$.) Suppose entry j of the samples is the indicator function for female samples. If, e.g., $c_j = -0.1$, then we are less likely to sample females, since whenever $(x_i)_j = 1$, this subtracts 0.1 from the log probability of sample i, giving the female samples a smaller chance of being sampled. On the other hand, if $c_j = 0.1$, then samples that are female get 0.1 added to their log probabilities. Next we take 10000 samples without replacement from the columns of F using the distribution π. This sample is highly skewed; for example, 41.2% of samples are female, whereas in the original dataset 54.8% samples are female. Therefore, we need to do sample weighting to make the sample representative.

6.1 Maximum entropy weighting

We first find the maximum entropy weighting of these 10000 samples such that the induced expected values match the desired expected values, i.e., we solve the problem described in Sect. 3.1. We solved the problem using our ADMM implementation, which took around 21 s. (Using a generic cone program solver like SCS, the default solver for CVXPY, took 19 min.) The resulting entropy of w was 8.59, whereas the entropy of the uniform distribution on $\{1, \dots, 10000\}$ is 9.21. The histogram of w is shown in Fig. 1 and appears to have a long right tail, meaning that some samples are considerably up-weighted.

To illustrate a use for these weights, we compare the distributions of other columns not included in the procedure to the true distribution across the entire dataset. We did this for the height and weight columns, and in Figs. 2 and 3, we plot the cumulative distribution function (CDF) of height and weight
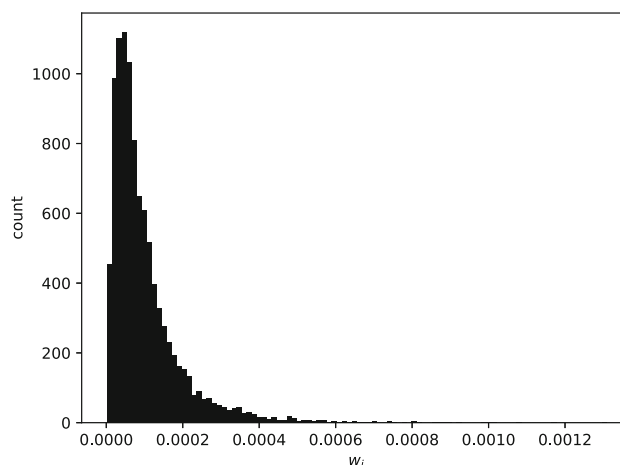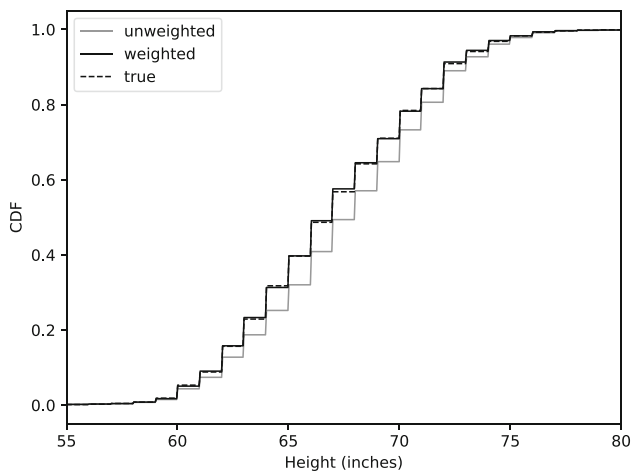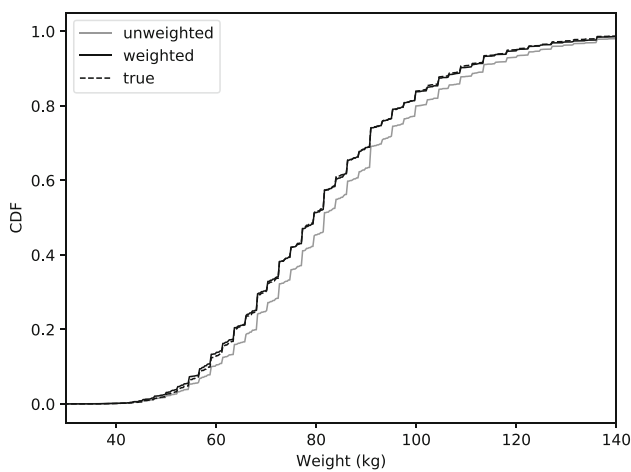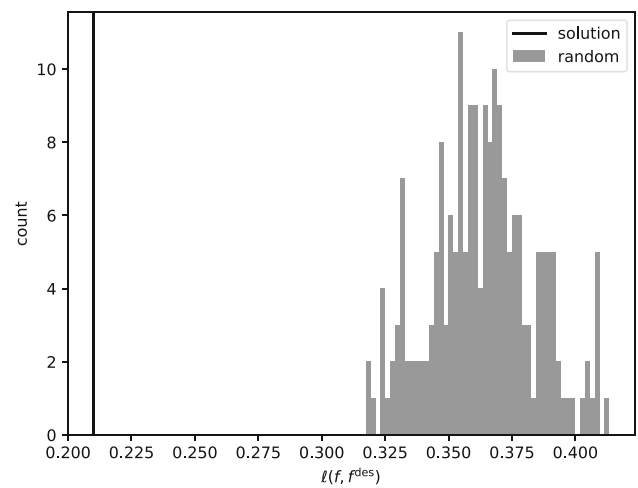

Fig. 1 Histogram of maximum entropy weights

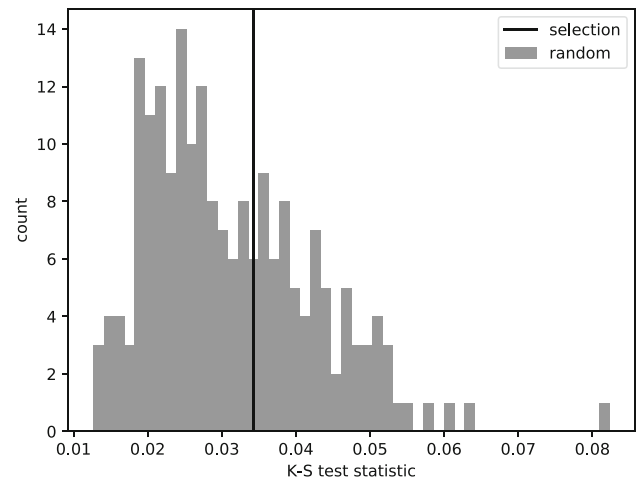**Fig. 2** CDF of height for unweighted, weighted, and true distributions

Fig. 3 CDF of weight for unweighted, weighted, and true distributions

Fig. 4 Histogram of losses for random subsamples and our selection (as a vertical line)

Fig. 5 Histogram of K-S test statistic for height

in the true distribution, and the unweighted and weighted distribution over the subsample. The weighted CDF appears to match the true CDF much better than the unweighted CDF. We computed the Kolmogorov–Smirnov (K-S) test statistic (Kolmogorov 1933), i.e., the maximum absolute difference between the CDFs, between the unweighted and true CDF and between the weighted and true CDF. For the height attribute, the K-S statistic was 0.078 for the unweighted distribution and 0.008 for the weighted distribution, which is almost a factor of 10 better. For the weight attribute, the K-S statistic was 0.064 for the unweighted distribution and 0.009 for the weighted distribution, a factor of more than 7 better.

We remind the reader that our focus here is not on using the weights, but only on how to compute them. We give these results only to show that weighting has indeed improved our estimate of the CDF of unrelated functions of the samples. We also note that a similar improvement in estimating the distribution of height and weight can be obtained with a far simpler representative sample weighting problem, for example, one that takes into account only sex and age group.

6.2 Representative selection

Next we consider the representative selection problem, described in Sect. 3.2, selecting $k = 500$ samples with uniform weights from the $n = 10000$ samples. Since we can no longer match f^{des} exactly, as the loss we use the sum of the KL divergence for each of state-age, sex, education-income, and general health. We approximately solved the problem using our ADMM implementation, initialized with a uniform distribution, which took under 12 s. The final loss of our weights was 0.21. We compared the loss to the loss of 200 random subsamples, which were each generated by sampling without replacement with the maximum entropy weights; the histogram of these objective values is displayed in Fig. 4. Our solution has substantially lower loss than even the best random subsample. We also computed the K-S statistic for the CDF of the height and weight columns between the 200 random samples and the true distribution, and between
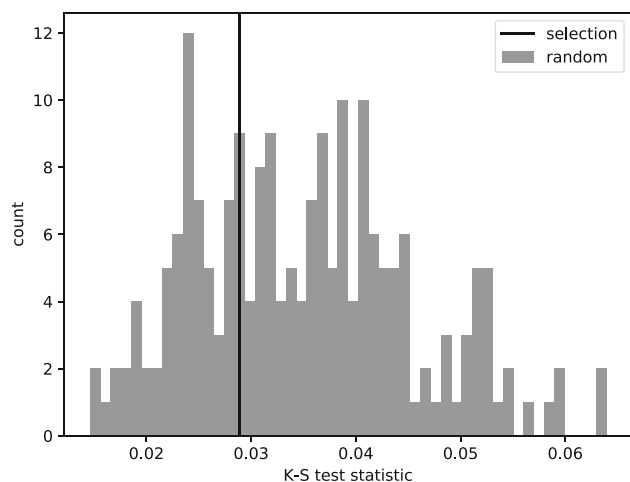
Fig. 6 Histogram of K-S test statistic for weight

our selection and the true distribution. The histogram of the test statistic, as well as our test statistic, for the height and weight CDFs is displayed in Figs. 5 and 6; we had a lower test statistic 36.5% of the time for height and 68.5% of the time for weight. So our selection is less representative in terms of height than if we had sampled using the maximum entropy weights, but more representative in terms of weight. (Both of these methods are far superior to selecting k samples uniformly at random from the 10000.)

7 Extensions and variations

In this section, we discuss extensions and variations.

Missing entries in some data Here, we describe what to do when some entries of F are missing, i.e., $F_{ij} \in \mathbf{R} \cup \{?\}$, where ? denotes a missing value. The simplest and often most effective thing to do when there are missing values is to replace them with the desired expected value for that function. That is, suppose one of our functions is the indicator function of a sample being female. If, for one sample, we did not know their sex, we would replace that missing value in F with the desired proportion of females. Assuming the loss is separable and homogeneous, which all aforementioned losses are, filling in missing entries this way has the same effect as ignoring the missing entries in the computation of $f - f^{\text{des}}$, and scaling the loss by some multiple of one minus the sum of the weights for samples with missing entries of that feature. In our implementation, we fill in missing entries this way.

Conditional probabilities Suppose F_1 is the $\{0, 1\}$ indicator function of some set $A \cap B$, where $A, B \subset \mathcal{X}$, so f_1 is the probability that $x \in A \cap B$, and F_2 is the $\{0, 1\}$ indicator function of B, so f_2 is the probability that $x \in B$. Then, the quantity f_1/f_2 is equal to the (conditional) probability that $x \in A$ given that $x \in B$. For example, if A contains the samples that are over the age of 65, and B contains the samples that are female, then f_1/f_2 is the probability that the

sample is over the age of 65 given it is female. If we want the induced probability f_1/f_2 to be close to some value f^{des}, we can use the loss

$$l(f, f^{\text{des}}) = l_0(f_1 - f_2 f^{\text{des}}),$$

where l_0 is some convex function. The function l is convex because $f_1 - f_2 f^{\text{des}}$ is affine in w.

CDFs Suppose $x \in \mathbf{R}$. We can make f the CDF evaluated at $\alpha_1, \ldots, \alpha_m \in \mathbf{R}$ by letting F_i be the $\{0,1\}$ indicator function of the set $\{x \mid x \leq \alpha_i\}$, $i = 1, \ldots, m$. Then f^{des} would be the desired CDF. A common loss function is the K-S test statistic, or

$$l(f, f^{\text{des}}) = \| f - f^{\text{des}} \|_\infty,$$

which is convex.

Compliance with ethical standards

Conflicts of interest Not applicable.

Code availability. All computer code is freely available online at www.github.com/cvxgrp/rsw.

A Iterative proportional fitting

The connection between iterative proportional fitting, initially proposed by Deming and Stephan (1940) and the maximum entropy weighting problem has long been known and has been explored by many authors Teh and Welling (2003), Fu et al. (2019), She and Tang (2019), Wittenberg (2010), Bishop et al. (2007). We provide a similar presentation to She and Tang (2019), Sect. 2.1, though we show that the iterative proportional fitting algorithm that is commonly implemented is actually a block coordinate descent algorithm on the dual variables, rather than a direct coordinate descent algorithm. Writing this update in terms of the primal variables gives exactly the usual iterative proportional fitting update over the marginal distribution of each property.

Maximum entropy problem In particular, we will analyze the application of block coordinate descent on the dual of the problem

$$\text{minimize} \quad \sum_{i=1}^{n} w_i \log w_i$$
$$\text{subject to} \quad Fw = f^{\text{des}} \tag{6}$$
$$\mathbf{1}^T w = 1, \quad w \geq 0,$$

with variable $w \in \mathbf{R}^n$, where the problem data matrix is Boolean, i.e., $F \in \{0, 1\}^{m \times n}$. This is just the maximum entropy weighting problem given in Sect. 3.1, but in the specific case where F is a matrix with Boolean entries.

Selector matrices We will assume that we have several possible categories $k = 1, \dots, N$ which the user has stratified over, and we will define selector matrices $S_k \in \{0, 1\}^{p_k \times m}$ which 'pick out' the rows of F containing the properties for property k. For example, if the first three rows of F specify the data entries corresponding to the first property, then S_1 would be a matrix such that

$$S_1 F = F_{1:3, 1:n},$$

and each column of $S_1 F$ is a unit vector, i.e., a vector whose entries are all zeros except at a single entry, where it is one. This is the same as saying that, for some property k, each data point is allowed to be in exactly one of the p_k possible classes. Additionally, since this should be a proper probability distribution, we will also require that $\mathbf{1}^T S_k f^{\text{des}} = 1$, i.e., the desired marginal distribution for property k should itself sum to 1.

Dual problem To show that iterative proportional fitting is equivalent to block coordinate ascent, we first formulate the dual problem (Boyd and Vandenberghe 2004, Ch. 5). The Lagrangian of (6) is

$$\mathcal{L}(w, \nu, \lambda) = \sum_{i=1}^{n} w_i \log w_i + \nu^T (Fw - f^{\text{des}}) + \lambda (\mathbf{1}^T w - 1),$$

where $\nu \in \mathbf{R}^n$ is the dual variable for the first constraint and $\lambda \in \mathbf{R}$ is the dual variable for the normalization constraint. Note that we do not need to include the nonnegativity constraint on w_i, since the domain of $w_i \log w_i$ is $w_i \geq 0$.

The dual function (Boyd and Vandenberghe 2004, Sect. 5.1.2) is given by

$$g(\nu, \lambda) = \inf_{w} \mathcal{L}(w, \nu, \lambda), \tag{7}$$

which is easily computed using the Fenchel conjugate of the negative entropy (Boyd and Vandenberghe 2004, Sect. 3.3.1):

$$g(\nu, \lambda) = -\mathbf{1}^T \exp(-(1 + \lambda)\mathbf{1} - F^T \nu) - \nu^T f^{\text{des}} - \lambda, \tag{8}$$

where exp of a vector is interpreted componentwise. Note that the optimal weights $w^\star$ are exactly those given by

$$w^\star = \exp(-(1 + \lambda)\mathbf{1} - F^T \nu). \tag{9}$$

Strong duality Because of strong duality, the maximum of the dual function (7) has the same value as the optimal value of the original problem (6) (Boyd and Vandenberghe 2004, Sect. 5.2.3). Because of this, it suffices to find an optimal pair of dual variables, λ and ν, which can then be used to find an optimal $w^\star$, via (9).

To do this, first partially maximize g with respect to λ, i.e.,

$$g^p(\nu) = \sup_{\lambda} g(\nu, \lambda).$$

We can find the minimum by differentiating (8) with respect to λ and setting the result to zero. This gives

$$1 + \lambda^\star = \log \left(\mathbf{1}^T \exp(-F^T \nu) \right),$$

while

$$g^p(\nu) = -\log(\mathbf{1}^T \exp(-F^T \nu)) - \nu^T f^{\text{des}}.$$

This also implies that, after using the optimal $\lambda^\star$ in (9),

$$w^\star = \frac{\exp(-F^T \nu)}{\mathbf{1}^T \exp(-F^T \nu)}. \tag{10}$$

Block coordinate ascent In order to maximize the dual function g^p, we will use the simple method of block coordinate ascent with respect to the dual variables corresponding to the constraints of each of the possible k categories. Equivalently, we will consider updates of the form

$$\nu^{t+1} = \nu^t + S_t^T \xi^t, \quad t = 1, \dots, T,$$

where ν^t is the dual variable at iteration t, while $\xi^t \in \mathbf{R}^{p_t}$ is the optimization variable we consider at iteration t. To simplify notation, we have used S_t to refer to the selector matrix at iteration t, if $t \leq N$, and otherwise set $S_t = S_{(t-1 \bmod N)+1}$, i.e., we choose the selector matrices in a round robin fashion. The updates result in an ascent algorithm, which is guaranteed to converge to the global optimum since g^p is a smooth concave function Tseng (2001).

Block coordinate update In order to apply the update rule to $g^p(\nu)$, we first work out the optimal steps defined as

$$\xi^t = \text{argmax}_\xi \ g^p(\nu^t + S_t^T \xi).$$

To do this, set the gradient of g_p to zero,

$$\nabla_\xi g^p(\nu^t + S_t^T \xi) = 0,$$

which implies that

$$\frac{\sum_{i=1}^{n} (S_t f_i) \exp(-f_i^T \nu^t - f_i^T S_t^T \xi)}{\sum_{i=1}^{n} \exp(-f_i^T \nu^t - f_i^T S_t^T \xi)} = S f^{\text{des}}, \tag{11}$$

where f_i is the ith column of F and the division is understood to be elementwise.

To simplify this expression, note that, for any unit basis vector $x \in \mathbf{R}^m$ (i.e., $x_i = 1$ for some i and 0, otherwise), we have the simple equality,

$$x \exp(x^T \xi) = x \circ \exp(\xi),$$

where $\circ$ indicates the elementwise product of two vectors. Using this result with $x = S_t f_i$ on each term of the numerator from the left hand side of (11) gives

$$\sum_{i=1}^{n} (S_t f_i) \exp(-f_i^T v^t - f_i^T S_t^T \xi) = \exp(-\xi) \circ y,$$

where $y = \sum_{i=1}^{m} \exp(-f_i^T v^t) S f_i$. We can then rewrite (11) in terms of y by multiplying the denominator on both sides of the expression:

$$\exp(-\xi) \circ y = (\exp(-\xi)^T y) S_t f^{\text{des}},$$

which implies that

$$\frac{y \circ \exp(-\xi)}{\mathbf{1}^T (y \circ \exp(-\xi))} = S_t f^{\text{des}}.$$

Since $\mathbf{1}^T S_t f^{\text{des}} = 1$,

$$y \circ \exp(-\xi) = S_t f^{\text{des}},$$

or, after solving for ξ,

$$\xi = - \log \left(\mathbf{diag}(y)^{-1} S_t f^{\text{des}} \right),$$

where the logarithm is taken elementwise. The resulting block coordinate ascent update can be written as

$$v^{t+1} = v^t - S_t^T \log \left(\frac{S_t f^{\text{des}}}{\sum_{i=1}^{n} \exp(-f_i^T v^t) S_t f_i} \right), \quad (12)$$

where the logarithm and division are performed elementwise. This update can be interpreted as changing v in the entries corresponding to the constraints given by property t by adding the log difference between the desired distribution and the (unnormalized) marginal distribution for this property suggested by the previous update. This follows from (10), which implies $w_i^t \propto \exp(-f_i^T v^t)$ for each $i = 1, \ldots, n$, where w^t is the distribution suggested by v^t at iteration t. *Resulting update over w* We can rewrite the update for the dual variables v as a multiplicative update for the primal variable w, which is exactly the update given by the iterative proportional fitting algorithm. More specifically, from (10),

$$w_i^{t+1} = \frac{\exp(-f_i^T v^{t+1})}{\sum_{i=1}^{n} \exp(-f_i^T v^{t+1})}.$$

For notational convenience, we will write $x_{ti} = S_t f_i$, which is a unit vector denoting the category to which data point i belongs to, for property t. Plugging update (12) gives, after some algebra,

$$\exp(-f_i^T v^{t+1}) = \exp(-f_i^T v^t) \frac{\exp(x_{ti}^T \log(S_t f^{\text{des}}))}{\exp\left(x_{ti}^T \log\left(\sum_j \exp(-f_j^T v^t) x_{tj}\right)\right)}.$$

Since x_{ti} is a unit vector, $\exp(x_{ti}^T \log(y)) = x_{ti}^T y$ for any vector $y > 0$, so

$$\exp(-f_i^T v^{t+1}) = \exp(-f_i^T v^t) \frac{x_{ti}^T S_t f^{\text{des}}}{\sum_{j=1}^{n} \exp(-f_j^T v^t) x_{ti}^T x_{tj}}.$$

Finally, using (10) with v^t gives

$$w_i^{t+1} = w_i^t \frac{x_{ti}^T S_t f^{\text{des}}}{\sum_{j=1}^{n} w_j^t (x_{ti}^T x_{tj})},$$

which is exactly the multiplicative update of the iterative proportional fitting algorithm, performed for property t.

B Expected values of BRFSS data

See Tables 1, 2, 3, and 4.

Table 1 Desired sex values in percentages

Male	Female
45.2	54.8

Table 2 Desired education and income values in percentages

Education	25–35k	35–50k	50–75k	70k+
Elementary	0.455	0.355	0.432	0.344
High school	1.877	2.059	3.064	3.702
Some college	1.172	1.440	2.098	2.8612
College	0.552	0.666	1.024	1.653
Elementary	0.266	0.168	0.084	0.096
High school	3.941	4.313	3.876	4.730
Some college	3.602	4.752	5.198	7.992
College	2.547	4.691	7.415	22.576

Table 3 Desired reported health values in percentages

Excellent	Very Good	Good	Fair	Poor
16.48	32.6	31.7	13.93	5.3

Table 4 Desired state and age values in percentages

State	18–24	25–34	35–44	45–54	55–65	65+
AK	0.040	0.069	0.083	0.091	0.147	0.200
AL	0.070	0.139	0.176	0.239	0.332	0.553
AR	0.039	0.085	0.099	0.152	0.240	0.610
AZ	0.095	0.184	0.195	0.259	0.374	0.741
CA	0.260	0.453	0.458	0.443	0.486	0.625
CO	0.127	0.243	0.262	0.310	0.410	0.683
CT	0.102	0.200	0.241	0.435	0.558	0.913
DC	0.042	0.107	0.128	0.166	0.188	0.374
DE	0.072	0.120	0.144	0.185	0.253	0.422
FL	0.201	0.367	0.369	0.509	0.657	1.382
GA	0.157	0.257	0.290	0.346	0.419	0.658
GU	0.035	0.051	0.055	0.077	0.076	0.081
HI	0.119	0.212	0.241	0.291	0.361	0.582
IA	0.148	0.228	0.280	0.330	0.426	0.673
ID	0.062	0.096	0.098	0.110	0.155	0.326
IL	0.097	0.163	0.171	0.203	0.235	0.344
IN	0.084	0.145	0.168	0.274	0.370	0.698
KS	0.151	0.249	0.275	0.371	0.503	0.924
KY	0.129	0.192	0.218	0.332	0.436	0.647
LA	0.083	0.150	0.157	0.204	0.256	0.322
MA	0.115	0.184	0.176	0.249	0.306	0.495
MD	0.147	0.294	0.386	0.646	0.884	1.655
ME	0.073	0.165	0.206	0.342	0.558	1.164
MI	0.187	0.280	0.284	0.374	0.488	0.746
MN	0.277	0.459	0.543	0.639	0.818	1.148
MO	0.064	0.128	0.137	0.185	0.283	0.627
MS	0.078	0.135	0.170	0.221	0.296	0.437
MT	0.066	0.113	0.140	0.158	0.261	0.449
NC	0.086	0.144	0.146	0.186	0.192	0.327
ND	0.058	0.108	0.123	0.166	0.289	0.541
NE	0.211	0.353	0.417	0.450	0.669	1.245
NH	0.039	0.081	0.119	0.202	0.319	0.551
NJ	0.048	0.095	0.106	0.115	0.139	0.203
NM	0.092	0.167	0.185	0.222	0.337	0.532
NV	0.058	0.095	0.092	0.115	0.135	0.242
NY	0.506	0.945	1.025	1.384	1.744	2.572
OH	0.151	0.263	0.290	0.438	0.668	1.108
OK	0.063	0.126	0.142	0.163	0.246	0.461
OR	0.097	0.179	0.192	0.233	0.250	0.408
PA	0.110	0.190	0.190	0.237	0.282	0.413
PR	0.080	0.151	0.165	0.198	0.213	0.300
RI	0.058	0.112	0.144	0.208	0.294	0.466
SC	0.102	0.181	0.239	0.332	0.503	1.122
SD	0.084	0.160	0.173	0.234	0.345	0.633
TN	0.069	0.139	0.156	0.191	0.248	0.377

Table 4 continued

State	18–24	25–34	35–44	45–54	55–65	65+
TX	0.168	0.332	0.327	0.373	0.463	0.902
UT	0.241	0.376	0.434	0.382	0.389	0.591
VA	0.133	0.251	0.285	0.395	0.511	0.783
VT	0.052	0.114	0.162	0.247	0.363	0.557
WA	0.155	0.329	0.364	0.415	0.590	1.146
WI	0.069	0.119	0.133	0.184	0.253	0.384
WV	0.050	0.091	0.128	0.179	0.253	0.419
WY	0.047	0.082	0.109	0.118	0.239	0.425

References

Agrawal, A., Verschueren, R., Diamond, S., Boyd, S.: A rewriting system for convex optimization problems. J. Control Decis. **5**(1), 42–60 (2018)

Angeris, G., Vučković, J., Boyd, S.: Computational bounds for photonic design. ACS Photonics **6**(5), 1232–1239 (2019). https://doi.org/10.1021/acsphotonics.9b00154. ISSN 2330-4022, 2330-4022

ApS, M.: MOSEK modeling cookbook. https://docs.mosek.com/MOSEKModelingCookbook.pdf (2020)

Bethlehem, J., Keller, W.: Linear weighting of sample survey data. J. Off. Stat. **3**(2), 141–153 (1987)

Bishop, Y., Fienberg, S., Holland, P.: Discrete Multivariate Analysis. Springer, New York (2007). 978-0-387-72805-6

Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004). 978-0-521-83378-3

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends® Mach. Learn. **3**(1), 1–122 (2010). https://doi.org/10.1561/2200000016. ISSN 1935-8237, 1935-8245

Center for Disease Control and Prevention (CDC). Behavioral Risk Factor Surveillance System Survey Data (2018a)

Center for Disease Control and Prevention (CDC). LLCP 2018 codebook report. https://www.cdc.gov/brfss/annual_data/2018/pdf/codebook18_llcp-v2-508.pdf (2018b)

Chen, S., Donoho, D., Saunders, M.: Atomic decomposition by basis pursuit. SIAM Rev. **43**(1), 129–159 (2001)

Daszykowski, M., Walczak, B., Massart, D.: Representative subset selection. Anal. Chim. Acta **468**(1), 91–103 (2002)

Deming, W., Stephan, F.: On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. Ann. Math. Stat. **11**(4), 427–444 (1940)

Deville, J.-C., Särndal, C.-E., Sautory, O.: Generalized raking procedures in survey sampling. J. Am. Stat. Assoc. **88**(423), 1013–1020 (1993)

Diamond, S., Boyd, S.: CVXPY: A Python-embedded modeling language for convex optimization. J. Mach. Learn. Res. **17**(83), 1–5 (2016)

Diamond, S., Takapoui, R., Boyd, S.: A general system for heuristic minimization of convex functions over non-convex sets. Optim. Methods Softw. **33**(1), 165–193 (2018)

Domahidi, A., Chu, E., Boyd, S.: ECOS: An SOCP solver for embedded systems. In 2013 European Control Conference (ECC), pp. 3071–3076, Zurich (2013). IEEE. ISBN 978-3-033-03962-9. https://doi.org/10.23919/ECC.2013.6669541

Estabrooks, A., Jo, T., Japkowicz, N.: A multiple resampling method for learning from imbalanced data sets. Comput. Intell. **20**(1), 18–36 (2004)

Fougner, C., Boyd, S.: Parameter selection and preconditioning for a graph form solver. Emerging Applications of Control and Systems Theory, pp. 41–61. Springer, Cham (2018)

Fu, A., Narasimhan, B., Boyd, S.: CVXR: an R package for disciplined convex optimization. J. Stat. Softw. **94**, 1–34 (2019)

Grant, M., Boyd, S.: Graph implementations for nonsmooth convex programs. Recent Advances in Learning and Control. Lecture Notes in Control and Information Sciences, pp. 95–110. Springer, London (2008)

Grant, M., Boyd, S.: CVX: Matlab software for disciplined convex programming, version 2.1 (2014)

Gurobi Optimization. GUROBI optimizer reference manual. https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/9.0/refman.pdf (2020)

Heckman, J.: The common structure of statistical models of truncation, sample selection and limited dependent variables and a simple estimator for such models. Ann. Econ. Soc. Meas. **5**, 475–492 (1976)

Holt, D., Smith, F.: Post stratification. J. R. Stat. Soc. Ser. A **142**(1), 33–46 (1979)

Horvitz, D., Thompson, D.: A generalization of sampling without replacement from a finite universe. J. Am. Stat. Assoc. **47**(260), 663–685 (1952)

Iannacchione, V., Milne, J., Folsom, R.: Response probability weight adjustments using logistic regression. Proc. Surv. Res. Methods Sect. Am. Stat. Assoc. **20**, 637–642 (1991)

Jones, E., Oliphant, T., Peterson, P.: SciPy: Open source scientific tools for Python. http://www.scipy.org/ (2001)

Kalton, G., Flores-Cervantes, I.: Weighting methods. J. Off. Stat. **19**(2), 81 (2003)

Karp, R.: Reducibility among combinatorial problems. Complexity of Computer Computations, pp. 85–103. Springer, Boston (1972). https://doi.org/10.1007/978-1-4684-2001-2_9. ISBN 978-1-4684-2003-6 978-1-4684-2001-2

Kish, L.: Weighting for unequal pi. J. Off. Stat. **8**(2), 183–200 (1992)

Kolmogorov, A.: Sulla determinazione empírica di uma legge di distribuzione (1933)

Kruithof, J.: Telefoonverkeersrekening. De Ingenieur **52**, 15–25 (1937)

Kullback, S., Leibler, R.: On information and sufficiency. Ann. Math. Stat. **22**(1), 79–86 (1951)

Lambert, J.: Observations variae in mathesin puram. Acta Helvitica, physico-mathematico-anatomico-botanico-medica **3**, 128–168 (1758)

Lavallée, P., Beaumont, J.-F.: Why we should put some weight on weights. In: Survey Methods, Insights from the Field (SMIF) (2015)

Lepkowski, J., Kalton, G., Kasprzyk, D.: Weighting adjustments for partial nonresponse in the 1984 SIPP panel. In Proceedings of the Section on Survey Research Methods, pp. 296–301. American Statistical Association Washington, DC, (1989)

Lofberg, J.: YALMIP: A toolbox for modeling and optimization in MATLAB. In: IEEE International Conference on Robotics and Automation, IEEE, pp. 284–289 (2004)

Lumley, T.: Complex surveys: a guide to analysis using R, vol. 565. Wiley, Hoboken (2011)

McKinney, W.: Data structures for statistical computing in Python. In: Proceedings of the 9th Python in Science Conference, pp. 56–61 (2010). https://doi.org/10.25080/Majora-92bf1922-00a

Mercer, A., Lau, A., Kennedy, C.: How different weighting methods work. https://www.pewresearch.org/methods/2018/01/26/how-different-weighting-methods-work/ (2018)

Neyman, J.: On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection. J. R. Stat. Soc. **96**(4), 558–625 (1934)

O'Donoghue, B., Chu, E., Parikh, N., Boyd, S.: Conic optimization via operator splitting and homogeneous self-dual embedding. J. Optim. Theory Appl. **169**(3), 1042–1068 (2016)

Parikh, N., Boyd, S.: `proximal` Github repository. https://github.com/cvxgrp/proximal (2013)

Parikh, N., Boyd, S.: Block splitting for distributed optimization. Math. Program. Comput. **6**(1), 77–102 (2014a)

Parikh, N., Boyd, S.: Proximal algorithms. Found. Trends® Optim. **1**(3), 127–239 (2014b). https://doi.org/10.1561/2400000003. ISSN 2167-3888, 2167-3918

Peyré, G., Cuturi, M.: Computational optimal transport: with applications to data science. Found. Trends® Mach. Learn. **11**(5–6), 355–607 (2019)

She, Y., Tang, S.: Iterative proportional scaling revisited: a modern optimization perspective. J. Comput. Graph. Stat. **28**(1), 48–60 (2019)

Stella, L., Antonello, N., Falt, M.: ProximalOperators.jl. https://github.com/kul-forbes/ProximalOperators.jl (2020)

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., Boyd, S.: qdldl: a free LDL factorization routine. https://github.com/oxfordcontrol/qdldl (2020a)

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., Boyd, S.: OSQP: An operator splitting solver for quadratic programs. Math. Program. Comput. **12**, 637–672 (2020b). https://doi.org/10.1007/s12532-020-00179-2

Teh, Y., Welling, M.: On improving the efficiency of the iterative proportional fitting procedure. In: AIStats (2003)

Tseng, P.: Convergence of a block coordinate descent method for nondifferentiable minimization. J. Optim. Theory Appl. **109**(3), 475–494 (2001). https://doi.org/10.1023/A:1017501703105. ISSN 0022-3239, 1573-2878

Udell, M., Mohan, K., Zeng, D., Hong, J., Diamond, S., Boyd, S.: Convex optimization in Julia. Workshop on High Performance Technical Computing in Dynamic Languages (2014)

Valliant, R., Dever, J., Kreuter, F.: Practical Tools for Designing and Weighting Survey Samples. Springer, New York (2013)

Vanderbei, R.: Symmetric quasidefinite matrices. SIAM J. Optim. **5**(1), 100–113 (1995)

Walt, S., Colbert, C., Varoquaux, G.: The NumPy array: a structure for efficient numerical computation. Comput. Sci. Eng. **13**(2), 22 (2011)

Wittenberg, M.: An introduction to maximum entropy and minimum cross-entropy estimation using stata. Stata J. Promot. Commun. Stat. Stata **10**(3), 315–330 (2010). https://doi.org/10.1177/1536867X1001000301. ISSN 1536-867X, 1536-8734

Yu, C.: Resampling methods: concepts, applications, and justification. Pract. Assess. Res. Eval. **8**(1), 19 (2002)

Yule, U.: On the methods of measuring association between two attributes. J. R. Stat. Soc. **75**(6), 579–652 (1912)