Journal of Artificial Intelligence Research 1 (2016) 1-15 Submitted 1/16; published 1/16

Elastic Prototype Classification

Cynthia Rudin RUDIN@MIT.EDU
Computer Science and Artificial Intelligence Laboratory and Sloan School, Massachusetts Institute of Technology

Prashan Wanigasekara PRASHANW@MIT.EDU
Electrical Engineering and Computer Science, Massachusetts Institute of Technology

Jacob Bien JBIEN@CORNELL.EDU
Department of Statistics, Cornell University

Abstract

An *elastic prototype* classifier has the benefit of having reasons for its predictions. For a new observation, an elastic prototype classifier will return not just a predicted class, but also a prototype (a nearby exemplar from a small set of learned prototypes) and the reasons why the current observation is similar to the prototype (a learned ball of influence around the prototype). For instance, given a new medical patient, an elastic prototype classifier may classify him as being at risk for a disease based on his proximity to a particularly typical patient for that disease, along only the symptoms relevant for diagnosing that disease. Having an elastic distance measure that varies between prototypes is advantageous, as it means that members of a class can be compared in a way that is specific to that particular class. We would want to compare cardiac medical patients only on aspects of health that affect the heart, whereas cancer patients would be compared only on aspects relating to cancer. In this work, we introduce a mathematical programming method for elastic prototype classification. We also show that the method can be useful as a visualization tool for understanding data.

1. Introduction

People often reason about a present case in terms of prototypical past cases. For instance, when we think of a movie genre, we might remember one or two typical movies to represent that genre. There is a subfield of cognitive science called prototype theory (Rosch 1973), indicating that humans think of a category with respect to particular members of that category. This type of thinking is pervasive, and used not only in the way people think, but the way they communicate and teach. The Case Method of teaching used commonly in medical schools, law schools, and business schools teaches students to reason about new cases based on generalizing knowledge from past cases taught in class. Firefighters are also taught to consider past cases in order to make decisions (Klein 1989), which is called *recognition-primed decision making*. The Superflex program (Madrigal & Knopp 2008) taught in schools to help children with social thinking uses prototypes; it relies on superhero characters with prototypical types of behaviors (e.g., Rock Brain, Glass Man,

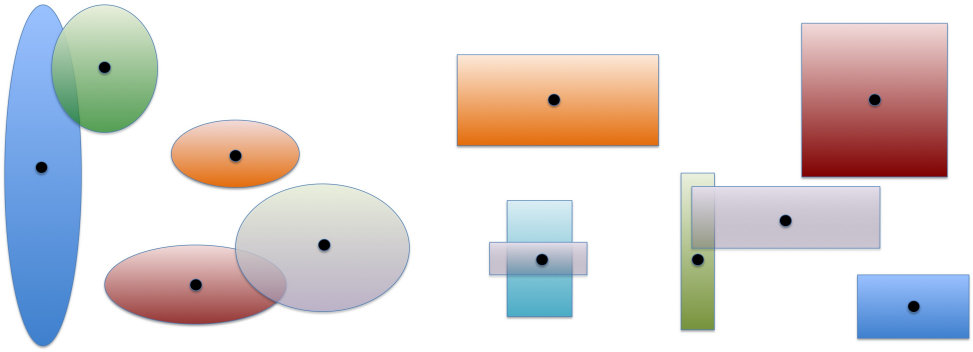


Figure 1: Illustrations of an elastic ellipsoidal prototype classifier (left), and an elastic rectangular prototype classifier (right). Balls can overlap and share a prototype.

etc.). More generally, case-based reasoning of various kinds (see Aamodt 1991, Slade 1991, Biswas et al. 2014) has been used in real-world examples for many years. Users of a system are generally more confident in a solution when an example case is shown, rather than simply the logic of the decision rule (Cunningham et al. 2003). Prototype and case-based reasoning are basic to how humans make tactical decisions (Newell & Simon 1972, Carroll 1980, Cohen et al. 1996, Klein 1989). In the setting of this work, we claim that humans not only consider prototypes, but also the logic of how a new case relates to a nearby prototype. The reasons for being close to a prototype are allowed to vary from one prototype to the next. This will capture the situation where, for instance, we should compare cardiac medical patients only on aspects of health that affect the heart, whereas cancer patients would be compared only on aspects relating to cancer.

In this work we develop a particular methodology for exemplar-based reasoning that we call *elastic prototype* classification. In elastic prototype classification, a small number of prototypes are used to represent the full data set. Each prototype represents a set of observations within a certain *ball of influence*. The selection of prototypes is learned from data, as well as the shape of the ball of influence. The ball of influence could be an ellipsoid or a hyper-rectangle, and we consider both in this paper. Illustrations of elastic prototype classifiers are provided in Figure 1, showing balls of influence that are ellipsoids and hyper-rectangles, and where the prototype is at the center of each ball. The ellipsoids could be elongated in certain directions. If the ellipsoid were highly elongated on a particular feature, it would mean that this feature is not relevant for the comparison (e.g., comparing cardiac patients on their food allergies is not important). If the ellipsoid were very narrow along a certain feature, it means that an example must have a very similar value of that feature to be within the prototype's ball of influence (e.g., blood pressure is important for a subset of cardiac patients). We have found in practice that allowing the ball of influence

to differ between prototypes is very important in terms of the model's ability to fit the data and to provide useful visualizations of the data.

Our learning algorithm is based on mixed-integer linear programming. We were able to create formulations for both the ellipsoidal and rectangular cases that are linear in all decision variables. We provide these formulations, and suggest alternating minimization schemes that help to speed up computation. When the data can be plotted in two or three dimensions, elastic prototype classifiers can also be useful as visualization tools. Each ball of influence can be displayed along with its prototype at the center, as in Figure 1.

Despite exponential improvements in the quality of MIP solvers such as CPLEX and Gurobi, many researchers choose not to use them anyway, possibly based on an (incorrect) 30-year old reputation of slowness. It is not clear why one would make an a priori assumption that branch-and-bound methods are not as useful as approximate methods such as simulated annealing or MCMC, gradient methods (e.g., backpropagation), or genetic algorithms that are commonly used for machine learning. MIP software packages are arguably the most principled way to solve these problems. This is because: (i) They provide a certificate of optimality. We can know with absolute certainty that we have reached an optimal solution. Other approximate methods (e.g., simulated annealing or MCMC, genetic algorithms, EM, etc.) could be far from optimal, and yet the user would never know that. (ii) Packages CPLEX and Gurobi have experienced an *exponential* speedup over the past 30 years. In 2010 CPLEX was 80K times faster than in 1991 on the same hardware. Considering how far hardware has also advanced, this is a massive speedup. (See slides of Bixby 2010, entitled "Mixed-Integer Programming: It works better than you may think"). (iii) It allows the user a level of flexibility and control over the problem that is not possible with other approaches. For instance, if there are many constraints, methods like simulated annealing (or MCMC) will generally not be able to find a feasible solution, let alone an optimal one. In order to help the solver work most effectively, we sure we have a mixed integer *linear* program, and that our formulation is as strong as possible.

2. Elastic Prototype Classification

We start with training data $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$ is the feature vector and $y_i \in \{1, \ldots, K\}$ is the class label (K classes in total) of the ith observation. An elastic prototype classifier consists of: (i) prototypes, which are a subset of the training data $\mathcal{P} \subseteq \{1, ..., n\}$, (ii) balls of influence around each $p \in \mathcal{P}$, which could be ellipsoids, represented by the length along each axis, or hyper-rectangles, represented by the lengths of each side. Our algorithm learns both the choice of prototypes and the shapes of the balls from the training data.

Each new case is classified according to which balls of influence it lies within. If it lies within one ball, it is assigned the class of that ball's prototype. If it lies within no balls, it is not assigned to a class. If it lies within multiple prototypes' balls, it could be assigned to multiple classes.

2.1 Related Work

A key to Elastic Prototype Classification's usefulness is that it blends prototype learning with other approaches that are commonly used for interpretable classification, such as feature selection and boolean logic. Prototype learning focuses on selecting a small number of representative examples from each class that can be used to summarize a much larger data set. Perhaps the simplest approach would be to apply an unsupervised prototype method, such as k-medoids (Kaufman & Rousseeuw 1990), to each class individually. However, more sophisticated prototype approaches use data from all classes to select each prototype, ensuring that the selected prototypes of a class avoid other classes' points. Bien & Tibshirani (2011) and Priebe et al. (2003) select a small set of spherical prototype balls for each class that avoid covering other classes. An obvious drawback to these methods is that non-spherical groups of points will require potentially large numbers of prototype balls to provide adequate coverage. This is because the balls cannot stretch, so multiple smaller balls are needed to cover the same area. In our experiments, the previous prototype methods tended to place a huge number of balls. A bigger drawback to using spherical balls is that this does not provide information about *why* a certain point is a prototype. For example, a doctor using the output of one of these methods might have to spend a lot of time comparing the selected prototype's medical history to that of the current patient to understand what they have in common; by contrast, an elastic ball would specifically highlight the few features that were relevant in selecting the prototype.

Another approach to interpretable classification uses decision trees to partition the feature space into hyper-rectangles (Breiman et al. 1984, Quinlan 1986, 1993, Kuhn et al. 2012). These methods do not involve prototypes, but are interpretable because the user can examine the set of steps used to arrive at the final decision. Our rectangular elastic balls bear a resemblance to the leaves of a decision tree, but are not a partition both because they can overlap with each other and because they do not cover the entire space.

A broad literature of papers uses feature selection to improve the interpretability (and performance) of classifiers. Our elastic balls can be thought of as providing prototype-local subspaces of feature space. This is related to methods that seek cluster-specific subspaces of feature space (Garg et al. 2015, Guan et al. 2011, Wang et al. 2015, 2013), but these do not use prototypes. A theoretical analysis on online pattern discovery (clustering with cluster-specific feature selection) is given by Huggins & Rudin (2014), again without prototypes. The method of Kim et al. (2014) uses prototypes, but for clustering, which is unsupervised. These methods that perform cluster-specific feature selection are reminiscent of bi-clustering methods, such as the plaid model (Lazzeroni & Owen 2002), in which observations are grouped according to subsets of features (though, again, this model does not involve learning prototypes).

We conclude by noting that the notion of set covering, which is central to elastic prototype classification, is a fundamental idea in mathematics and thus appears throughout machine learning and related fields (Tipping & Schölkopf 2001, Marchand & Taylor 2003).

Most relevant to the present work, Efrat et al. (2004) consider covering a set of points with a small number of axis-aligned ellipses while avoiding a certain set of "forbidden" points. In contrast to elastic prototype balls, their ellipses are not centered on actual observations from the data set, nor is their purpose to identify prototypical elements. Likewise, Bereg et al. (2012) seek rectangles that cover one class while avoiding another, but also do not require these rectangles to be centered at actual observations. Other methods of a similar flavor are those of Goh & Rudin (2014), Malioutov & Varshney (2013), Friedman & Fisher (1999) which use hyper-rectangles.

2.2 Specifications

Ideally, we have several goals in choosing the prototypes and their balls of influence:

- Classify training points correctly, by placing them in balls whose prototype has the correct class.
- Try not to misclassify training points, which happens when we place them in balls whose prototype has the wrong class.
- Use a small number of balls.
- Use an even smaller number of prototypes, which means we would prefer to use the same prototype for multiple balls of different shapes rather than having distinct prototypes.
- Constrain the ball in only a small number of dimensions. If the ball extends infinitely in some dimensions, then those dimensions are not the important features characterizing the ball. We would like each ball to have a few important features, so its size and shape is easier to communicate to users. In other words, we would like to have balls that extend infinitely in as many directions as possible.
- Involve as few dimensions as possible in defining the classifier (by reusing dimensions as much as possible across different balls).

Let us start with the formulation that uses ellipsoids for the balls of influence.

3. Ellipsoids Formulation

The prototypes and balls of influence will be learned using a machine learning algorithm based on mixed-integer linear programming. We first provide the terminology and then the formulation.

3.1 Definitions

- We observe n pairs $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^p$ and $y_i \in \{1, \ldots, K\}$. This is multiclass classification with K classes.

- Let $N_k = \{i : y_i = k\}$, the number of observations in class k.

- There are at most B balls. A natural choice is $B = n$, but one could also choose a smaller B if it is desired to limit the number of balls.

- Let $B(x, \Omega)$ denote an ellipsoid centered at $x \in \mathbb{R}^p$ and with shape determined by a $p \times p$ diagonal, positive definite matrix Ω:

$$B(x, \Omega) = \{x' \in \mathbb{R}^p : (x' - x)^T \Omega (x' - x) \leq 1\}.$$

  Bounds on the eigenvalues $\omega_1 I_p \preceq \Omega \preceq \omega_2 I_p$ (for $0 \leq \omega_1 \leq \omega_2 < \infty$) would enforce that no point in the ellipsoid is farther (in unsquared Euclidean distance) than $1/\sqrt{\omega_1}$ from the prototype and that a point outside of the ellipsoid is at least $1/\sqrt{\omega_2}$ from the prototype. The choice $\omega_1 = 0$ allows for ellipsoids that extend infinitely in certain directions. Having many zeros in Ω is good from an interpretability standpoint: Suppose all rows and columns are zero except for the first two. That leaves the submatrix $\Omega_{1:2,1:2} \in \mathbb{R}^{2\times 2}$ as the only nonzero part. If $B_2(x_{1:2}, \Omega_{1:2,1:2})$ is the two-dimensional ball defined by this submatrix, then our full p-dimensional "ball" is actually a cylinder (with an ellipsoid as cross-section) extending infinitely in $p - 2$ directions: $B_2(x_{1:2}, \Omega_{1:2,1:2}) \times \mathbb{R}^{p-2}$. This ball is described simply as a restriction on the first two variables. In this work we choose axis-aligned ellipses so that Ω becomes diagonal, each eigenvector corresponds to one of the features.

- Ball b is written $B(x_j, \Omega^{(b)})$ where x_j is the feature vector for one of the n points. If x_j is the center of ball b, then we say that this ball has class y_j. The center point of a ball is called its prototype.

- α_{jb} is a binary indicator variable indicating whether x_j is ball b's prototype. Each ball has only one prototype; however, x_j can be the prototype for multiple balls.

- β_j is a binary indicator variable indicating whether x_j is a prototype of any balls. We will regularize this directly to reduce the total number of prototypes used.

- ξ_i is a binary indicator variable indicating whether point i is *not* covered by a ball of class y_i. This is considered an error (in particular, a false negative).

- η_i is a binary indicator variable indicating whether point i is covered by one or more balls of classes other than y_i. This is considered an error (in particular, a false positive).

- d_{ijb} is a real-valued distance between i and j according to ball b's distance metric.
- c_{ijb} is a binary indicator variable indicating that point i is covered by ball b and that ball b's prototype is x_j.
- We include a sparsity penalty on the matrix Ω. A simple choice is the elementwise ℓ_1 norm: $\sum_\ell |\Omega_{\ell,\ell}|$.
- v_ℓ is a binary indicator variable indicating whether $\Omega^{(b)}_{\ell,\ell}$ is nonzero for ℓ and any ball b. This means that the collection of balls is using dimension ℓ. We will regularize this to reduce the number of dimensions used.

3.2 Formulation

The first term in the objective below counts the number of errors, where an error is made either if an observation is not covered by a ball of the correct class or if an observation is covered by at least one ball of the wrong class. There can be up to two errors made for a single observation. The second term in the objective regularizes the total number of balls. By the constraints in the model (and by the fact that we are minimizing over α's), the value of $\sum_j \alpha_{jb}$ will be 1 if index b represents a ball or 0 if index b is not being used for a ball. This means $\sum_b \sum_j \alpha_{jb}$ simply counts the number of balls. The third term minimizes the total number of prototypes, so that we encourage balls of different shapes to have the same prototype when possible. The fourth term encourage balls to be defined by a small number of dimensions. The fifth term encourages the total number of dimensions used for all the balls to be small. The constants $C_\alpha, C_\beta, C_\Omega$ and C_v control the relative importance of each of the terms in the objective.

The key decision variables are the α_{jb}'s that tell us the center of each ball, and the $\Omega^{(b)}$'s that define the distance metric for each ball.

$$\min_{\alpha_{jb},\xi_i,\eta_i,\beta_j,\Omega^{(b)},v_\ell} \sum_{i=1}^{n}(\xi_i+\eta_i) + C_\alpha \sum_{j=1}^{n}\sum_{b=1}^{B}\alpha_{jb} + C_\beta \sum_{j=1}^{n}\beta_j + C_\Omega \sum_{b=1}^{B}\sum_{\ell}|\Omega^{(b)}_{\ell,\ell}| + C_v \sum_{\ell} v_\ell$$

subject to

$\Omega^{(b)} = \text{Diag}(p \times p)$

$\omega_1 I_p \preceq \Omega^{(b)} \preceq \omega_2 I_p$

(1) For each ball $b = 1, \ldots, B$: $\sum_{j=1}^{n} \alpha_{jb} \leq 1$ (No more than 1 prototype per ball)

(2) For each point $i = 1, \ldots, n$: $\sum_{b=1}^{B} \sum_{j \in N_{y_i}} c_{ijb} \geq 1 - \xi_i$ (Nearly all points correctly covered)

(3) For each point $i = 1, \ldots, n$: $\sum_{b=1}^{B} \sum_{j \notin N_{y_i}} c_{ijb} \leq B\eta_i$ (Nearly all points not wrongly covered)

(4) For each j : $\frac{1}{B} \sum_{b} \alpha_{jb} \leq \beta_j$ (Define β_j)

(5) For each i, j, b : $d_{ijb} = (x_i - x_j)^T \Omega^{(b)} (x_i - x_j)$ (Define d_{ijb} for convenience in notation)

(6) For each i, j, b : $\alpha_{jb} - d_{ijb} \leq c_{ijb}$
(If $\alpha_{jb} = 1$ and $d_{ijb} \leq 1 - \epsilon$ then $c_{ijb} = 1$. If $\alpha_{jb} = 0$ or $d_{ijb} \geq 1$ do nothing.)

(7) For each i, j, b : $c_{ijb} \leq \alpha_{jb}$ (If $\alpha_{jb} = 0$ then $c_{ijb} = 0$.)

(8) For each i, j, b : $d_{ijb} - 1 + \epsilon \leq M(1 - c_{ijb})$ (If $d_{ijb} \geq 1$ then $c_{ijb} = 0$)

(9) For each b, ℓ : $\Omega^{(b)}_{\ell,\ell} - \omega_1 \leq v_\ell \omega_2$ (Tells us when we are using direction ℓ.)

$\alpha_{jb}, \xi_i, \eta_i, \beta_j, c_{ijb}, v_\ell \in \{0, 1\}$.

All constraints and the objective are *linear* in the decision variables. The first five constraints are explained inline, defining α_{jb}, ξ_i, η_i, β_j and d_{ijb}. The sixth constraint is more complicated, defining c_{ijb}. It says that if prototype j is the center of ball b, and if point i is within ball b (meaning distance at most $1 - \epsilon$ from prototype j), then c_{ijb} is 1. The seventh and eighth constraints also help to define c_{ijb}. The ninth constraint defines v_ℓ as being 1 when the $\Omega^{(b)}_{\ell,\ell}$ values are bigger than ω_1. Recall that if $\Omega^{(b)}_{\ell,\ell}$ is ω_1, then the ℓth dimension is not being used. (The ball is extended to its maximum allowed size along that direction.)

The formulation presented above is not the only possible formulation for this problem. We developed several formulations with different sets of variables, this one having the best practical performance. Generally speaking, the quality of a formulation can sometimes result in enormous run-time benefits.

3.3 Restricting the size of balls

As background, the eigenvalue bounds $\omega_1 I_p \preceq \Omega \preceq \omega_2 I_p$ (for $0 \leq \omega_1 \leq \omega_2 < \infty$) prescribe that the length of the longest axis of the ellipsoid is $2/\sqrt{\omega_1}$. The length of the shortest axis is $2/\sqrt{\omega_2}$. To see this, note that for any v,

$$\omega_1 \|v\|_2^2 \leq v^T \Omega v \leq \omega_2 \|v\|_2^2.$$

If $x' \in B(x, \Omega)$, then

$$(x' - x)^T \Omega (x' - x) \leq 1,$$

which implies that $\omega_1 \|x' - x\|_2^2 \leq 1$. Thus,

$$x' \in B(x, \Omega) \implies \|x' - x\|_2 \leq \frac{1}{\sqrt{\omega_1}}.$$

If $x' \notin B(x, \Omega)$, then

$$(x' - x)^T \Omega (x' - x) > 1,$$

which implies that $\omega_2 \|x' - x\|^2 > 1$. Thus,

$$x' \notin B(x, \Omega) \implies \|x' - x\|_2 > \frac{1}{\sqrt{\omega}_2}.$$

This tells us that the major axis of our ellipsoid is at most $2/\sqrt{\omega_1}$ and the minor axis of our ellipsoid is at least $2/\sqrt{\omega_2}$.

3.4 Alternating Minimization Scheme

The formulation above can be used as written, though it helps to warm-start it with a good solution that can be obtained quickly. As an alternative, or in conjunction with the original formulation, we propose here an alternating scheme that often speeds up convergence. The alternating minimization scheme alternates between simpler versions of the ellipsoid formulation above.

Step 0 Choose many random prototypes (i.e., initialize α_{jb} and set β_j).

Step 1 Run the *Choose Subspace* mathematical program. *Choose Subspace* is a simplified version of the ellipsoid formulation above, where decision variables α_{jb} and β_j are fixed. This fixes the prototypes and centers of balls, and leaves the subspaces as decision variables. Thus constraints (1) and (4) are deterministic (contain no decision variables) and thus not used. The objective and all other constraints remain the same.

Step 2 Run the *Choose Prototypes* mathematical program. *Choose Prototypes* is a simplified version of the ellipsoid formulation above, where $\Omega^{(b)}_{\ell,\ell}$ and v_ℓ are fixed, and thus constraints (5) and (9) are deterministic (contain no decision variables). The objective and all other constraints remain the same. The formulation is warm-started using the current set of prototypes from Step 1. This determines α_{jb} and β_j. Go back to Step 1, fixing α_{jb} and β_j.

This scheme does not need to be adhered to strictly. At any point we can switch to the full ellipsoidal formulation, using the alternating minimization scheme as a warm start.

We will show results from this formulation after introducing the hyper-rectangles formulation.

4. Hyper-Rectangles Formulation

The notation is similar for the hyper-rectangles formulation, though the formulation is substantially different.

4.1 Definitions

- We observe n pairs $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^p$ and $y_i \in \{1, \ldots, K\}$.
- Let $N_k = \{i : y_i = k\}$.
- There are at most B balls. A natural choice is $B = n$, but one could also choose a smaller B.
- Let $B(x, \omega)$ denote a rectangle centered at $x \in \mathbb{R}^p$ and with side lengths given by $r \in \mathbb{R}^p$, where componentwise $r_\ell = 1/\omega_\ell$:

$$B(x, \omega) = \{x' \in \mathbb{R}^p : \omega_\ell |x'_\ell - x_\ell| \leq 1 \text{ for all } \ell\} = \prod_{\ell=1}^p [x_\ell - r_\ell, x_\ell + r_\ell].$$

- Ball b is written $B(x_j, \omega^{(b)})$ where x_j is one of the n points. If x_j is the center of ball b, then we say that this ball has class y_j. The center point of a ball is its prototype.
- α_{jb} indicates whether x_j is ball b's prototype. Each ball has only one prototype; however, x_j can be the prototype for multiple balls.
- β_j indicates whether x_j is a prototype of any balls.
- ξ_i indicates whether point i is *not* covered by a ball of class y_i.
- η_i indicates whether point i is covered by one or more balls of classes other than y_i.
- $d_{ib} = \max_\ell \omega_\ell^{(b)} |x_{i\ell} - x_{j\ell}|$ where j obeys $\alpha_{jb} = 1$. That is, to calculate the distance from the observation to the prototype, we would compute the observation's one-dimensional Euclidean distance from the prototype along each dimension. The quantity $\omega_\ell^{(b)} |x_{i\ell} - x_{j\ell}|$ is the fraction of the rectangle's half-side-length (in dimension ℓ) that is taken up by the Euclidean distance $|x_{i\ell} - x_{j\ell}|$. The distance d_{ib} is taken as the largest of these fractions, among all of the coordinates.
- $\zeta_{ibl} = \omega_\ell^{(b)} |x_{i\ell} - x_{j\ell}|$ which is used in the definition of the distance.
- c_{ijb} indicates that point i is covered by ball b and that ball b's prototype is x_j.
- We include a sparsity penalty on the vector ω. We choose the simple ℓ_1 norm: $\sum_\ell |\omega_\ell|$. Note that if $\omega_\ell = 0$, then the rectangular ball extends infinitely in the ℓth direction, meaning that feature ℓ is not needed to determine whether points lie in the ball. By contrast, a high value of ω_ℓ says that the ball is very sensitive to the ℓth feature, requiring that all points lie within $1/\omega_\ell$ of the prototype.
- v_ℓ indicates whether any $\omega_\ell^{(b)}$ is nonzero for any ball b. This means that the direction ℓ is actively being used.

4.2 Formulation

The first term in the objective below counts the number of errors, where an error is made if an observation is not covered by a ball of the correct class, and an error is made if an observation is covered by at least one ball of the wrong class. There can be up to two errors made for a single observation. The second term regularizes the number of balls, since again $\sum_b \sum_j \alpha_{jb}$ counts the number of balls. The third term minimizes the total number of prototypes, so that we encourage balls of different shapes to have the same prototype when possible. The fourth term is the sparsity penalty on ω, and the fifth term encourages the total number of dimensions for all the balls to be small. The key variables are the α_{jb}'s that tell us the center of each ball, and the vectors $\omega^{(b)}$ that define the distance metric for each ball.

$$\min_{\xi_i,\eta_i,\alpha_{jb},\beta_j,\omega_\ell^{(b)},v_\ell} \sum_{i=1}^{n}(\xi_i+\eta_i) + C_\alpha \sum_{j=1}^{n}\sum_{b=1}^{B}\alpha_{jb} + C_\beta \sum_{j=1}^{n}\beta_j + \frac{C_\omega}{\bar{\omega}}\sum_{b=1}^{B}\sum_{\ell}\omega_\ell^{(b)} + C_v\sum_{\ell}v_\ell$$

subject to

(1) For all ℓ, b : $\quad 0 \le \omega_\ell^{(b)} \le \bar{\omega}$

(2) For each ball $b = 1,\ldots,B$: $\quad \sum_{j=1}^{n}\alpha_{jb} \le 1$ (No more than 1 prototype per ball)

(3) For each point $i = 1,\ldots,n$: $\quad \sum_{b=1}^{B}\sum_{j\in N_{y_i}} c_{ijb} \ge 1-\xi_i$ (Nearly all points correctly covered)

(4) For each point $i = 1,\ldots,n$: $\quad \sum_{b=1}^{B}\sum_{j\notin N_{y_i}} c_{ijb} \le B\eta_i$ (Nearly all points not wrongly covered)

(5) For each j : $\quad \frac{1}{B}\sum_{b}\alpha_{jb} \le \beta_j$ (Define β_j)

(6) For each i,j,b,ℓ : $\quad \zeta_{ibl} \ge \omega_\ell^{(b)}|x_{i\ell}-x_{j\ell}| - M(1-\alpha_{jb})$ (Define $\zeta_{ibl} = \omega_\ell^{(b)}|x_{i\ell}-x_{j\ell}|$)

(7) For each i,j,b,ℓ : $\quad \zeta_{ibl} \le \omega_\ell^{(b)}|x_{i\ell}-x_{j\ell}| + M(1-\alpha_{jb})$ (Define $\zeta_{ibl} = \omega_\ell^{(b)}|x_{i\ell}-x_{j\ell}|$)

(8) For each i,b,ℓ : $\quad d_{ib} \le \zeta_{ib\ell} + M(1-\gamma_{ib\ell})$ $\quad$ (Defines $d_{ib} = \max_l(\zeta_{ib\ell})$)

(9) $\sum_l \gamma_{ib\ell} = 1\ \forall i\ \forall b$ $\quad$ (Defines $d_{ib} = \max_l(\zeta_{ib\ell})$)

(10) For each i,b,l : $\quad d_{ib} \ge \zeta_{ib\ell}$ $\quad$ (Defines $d_{ib} = \max_l(\zeta_{ib\ell})$)

(11) For each i,j,b : $\quad \alpha_{jb} - d_{ib} \le c_{ijb}$

(If $\alpha_{jb} = 1$ and $d_{ib} \le 1-\epsilon$ then $c_{ijb} = 1$. If $\alpha_{jb} = 0$ or $d_{ib} \ge 1$ do nothing.)

(12) For each i, j, b : $\quad c_{ijb} \leq \alpha_{jb}$ (If $\alpha_{jb} = 0$ then $c_{ijb} = 0$)

(13) For each i, j, b : $\quad d_{ib} - 1 + \epsilon \leq M(1 - c_{ijb})$ (If $d_{ib} \geq 1$ then $c_{ijb} = 0$)

(14) For each b : $\frac{\omega_\ell^{(b)}}{\bar{\omega}} \leq v_\ell$ (Defines v_ℓ to indicate when we are using direction ℓ)

$\xi_i, \eta_i, \alpha_{jb}, \beta_j, c_{ijb}, \gamma_{ib\ell}, v_\ell \in \{0, 1\}$.

Again all constraints and the objective are *linear* in the decision variables. The first five constraints are explained inline, defining α_{jb}, ξ_i, η_i, and β_j. Constraints (6)-(10) define d_{ijb}. Constraints (6) and (7) say that if $\alpha_{jb} = 1$ (we are using ball b and its prototype is j), then we must set $\zeta_{ib\ell}$ to be the scaled distance in the ℓth direction between x_i and the prototype. Constraints (8)-(10) set d_{ib} to be the maximum of the $\zeta_{ib\ell}$'s. This corresponds to the dimension with the largest fraction of distance from the prototype. Constraints (11)-(13) are identical to those in the ellipsoid formulation for defining c_{ijb}. Constraint (14) says that when $w_\ell > 0$, this dimension ℓ is being used, and thus v_ℓ should be set to 1.

Again this formulation was not the first one we found, there are many other possible formulations.

4.3 Alternating Minimization Scheme

The following alternating minimization scheme alternates between simpler versions of the hyper-rectangles formulation above.

Step 0 Choose many random prototypes (i.e., initialize α_{jb} and set β_j).

Step 1 Run the *Choose Subspace* mathematical program. *Choose Subspace* is a simplified version of the hyper-rectangles formulation above, where decision variables α_{jb} and β_j are fixed. This fixes the prototypes and centers of balls, and leaves the subspaces as decision variables. Thus constraints (2) and (5) are deterministic (contain no decision variables) and thus not used. The objective and all other constraints remain the same.

Step 2 Run the *Choose Prototypes* mathematical program. *Choose Prototypes* is a simplified version of the hyper-rectangles formulation, where $\omega_\ell^{(b)}$ and v_ℓ are fixed, and thus constraints (1) and (14) are deterministic (contain no decision variables). The objective and all other constraints remain the same. The formulation is warm-started using the current set of prototypes from Step 1. This determines α_{jb} and β_j. Go back to Step 1, fixing α_{jb} and β_j.

This scheme does not need to be adhered to strictly. At any point we can switch to the full hyper-rectangle formulation, using the alternating minimization scheme as a warm start.

4.4 Hierarchical Warm-Start

We tried the following procedure for warm-starting the rectangular method.[1] This method could be a useful alternative to the alternating minimization scheme. From the training set we randomly selected 100 points, and generated solutions. This was repeated 10 times and we chose the best of the warm start solutions to be a warm start for the procedure using 200 randomly chosen points. We repeated the choice of 200 points and the solution of the model 5 times. After we have these 5 solutions, we used all of them to warm start the solver on the full training set. We chose the best of the solutions to use as the final trained model.

5. Results

We illustrate first the main purpose of these tools, which is for interpreting data and visualization.

We used the USDA National Nutrient Database for Standard Reference, from the U.S. Department of Agriculture (U.S. Department of Agriculture, Agricultural Research Service 2013). This database is the standard reference database for nutritional content of food in the United States.

We considered three particularly interesting food categories that have substantial overlap in nutritional content, namely *cereals*, *snacks*, and *sweets*. For instance, cereals and snacks can sometimes be very sweet, and sometimes not. The database includes 873 food items in these three categories. Out of the 500 items, 221 were *cereals*, 75 were *snacks*, and 204 were *sweets*. We considered each food item's cholesterol content, sugar content, and iron content. The nutritional content is reported in normalized units relative to the mean.

5.1 Visualization

Using ellipsoids, Figure 2 shows how this particular example visually highlights the differences in nutritional content between these categories that are not obvious, for instance, that sweets have very low iron levels but a range of cholesterol levels and sugar levels, whereas cereals tend to be low cholesterol, but with a broad range in sugar and iron content (cereals are often iron fortified). Figure 3 shows rotated views of the same model, and Figure 4 shows each prototype and ball on its own.

Analogous plots for the hyper-rectangles are provided in Figure 5, Figure 6, and Figure 7. Figure 5 shows the full set of hyper-rectangles, Figure 6 shows rotated views, and Figure 7 shows each rectangle separately.

Table 1 provides a description of the prototypes, their balls (showing which dimensions were important for each ball), the number of points correctly and incorrectly covered by each ball, and the number of points in the class of the prototype but not covered by a ball

1. Thank you to Raymond Patterson from the University of Alberta for suggesting this.

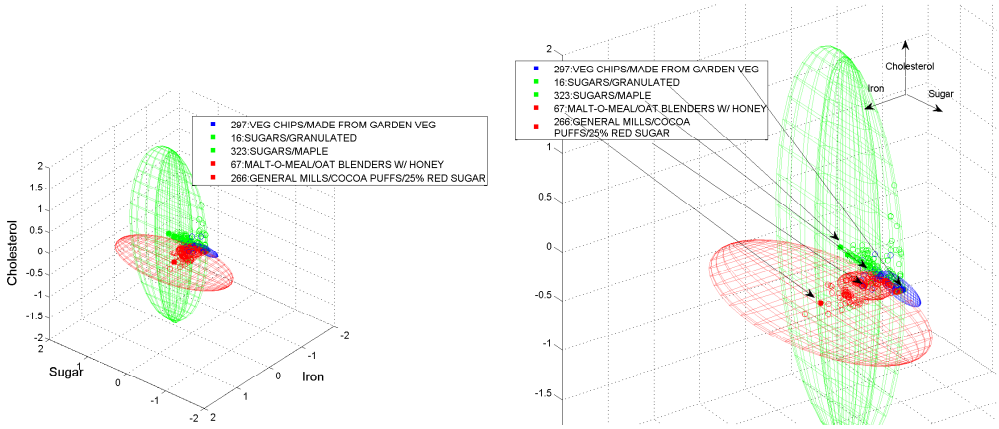
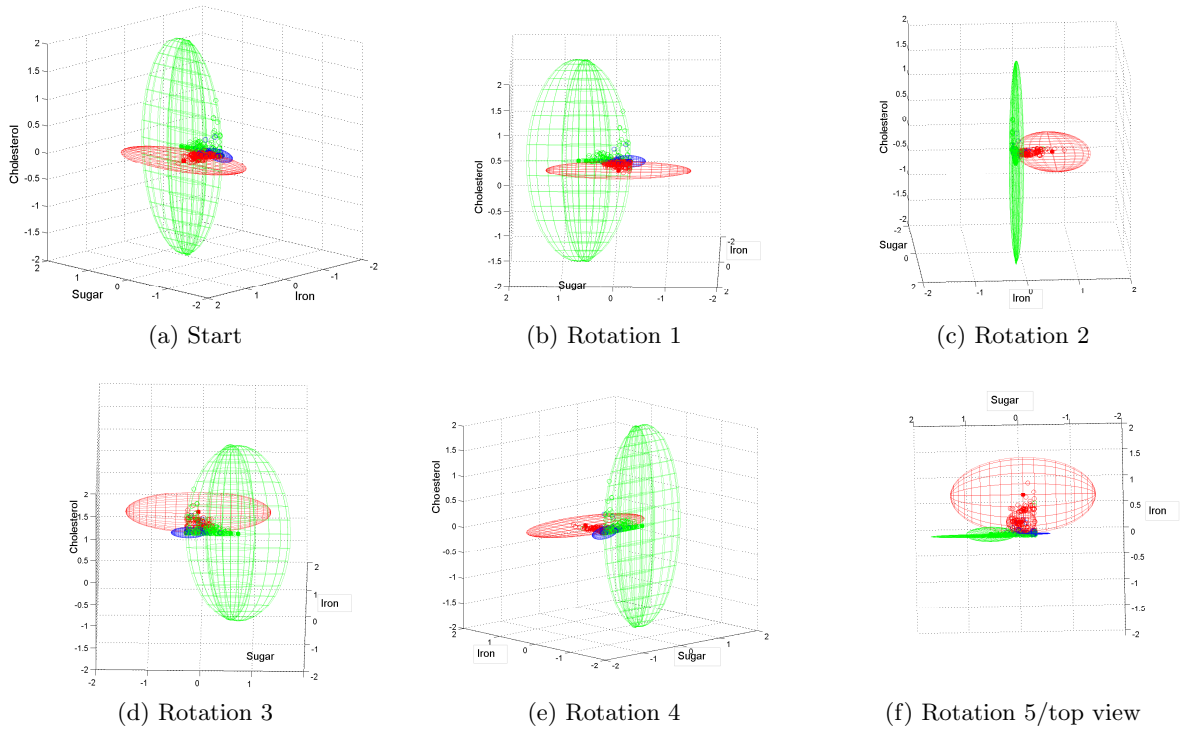


Figure 2: Ellipsoidal Elastic Prototype Classifier on the nutrition data set. There are 5 balls, shown from two different views. The prototypes are labeled in the right plot. The balls are colored according to the label of the ball's prototype. Sweets are in green, snacks are in blue, cereals are in red.



(a) Start (b) Rotation 1 (c) Rotation 2

(d) Rotation 3 (e) Rotation 4 (f) Rotation 5/top view

Figure 3: Rotated views of the ellipsoid classifier for the nutrition data set. Red: Cereals, Blue: Snacks, Green: Sweets

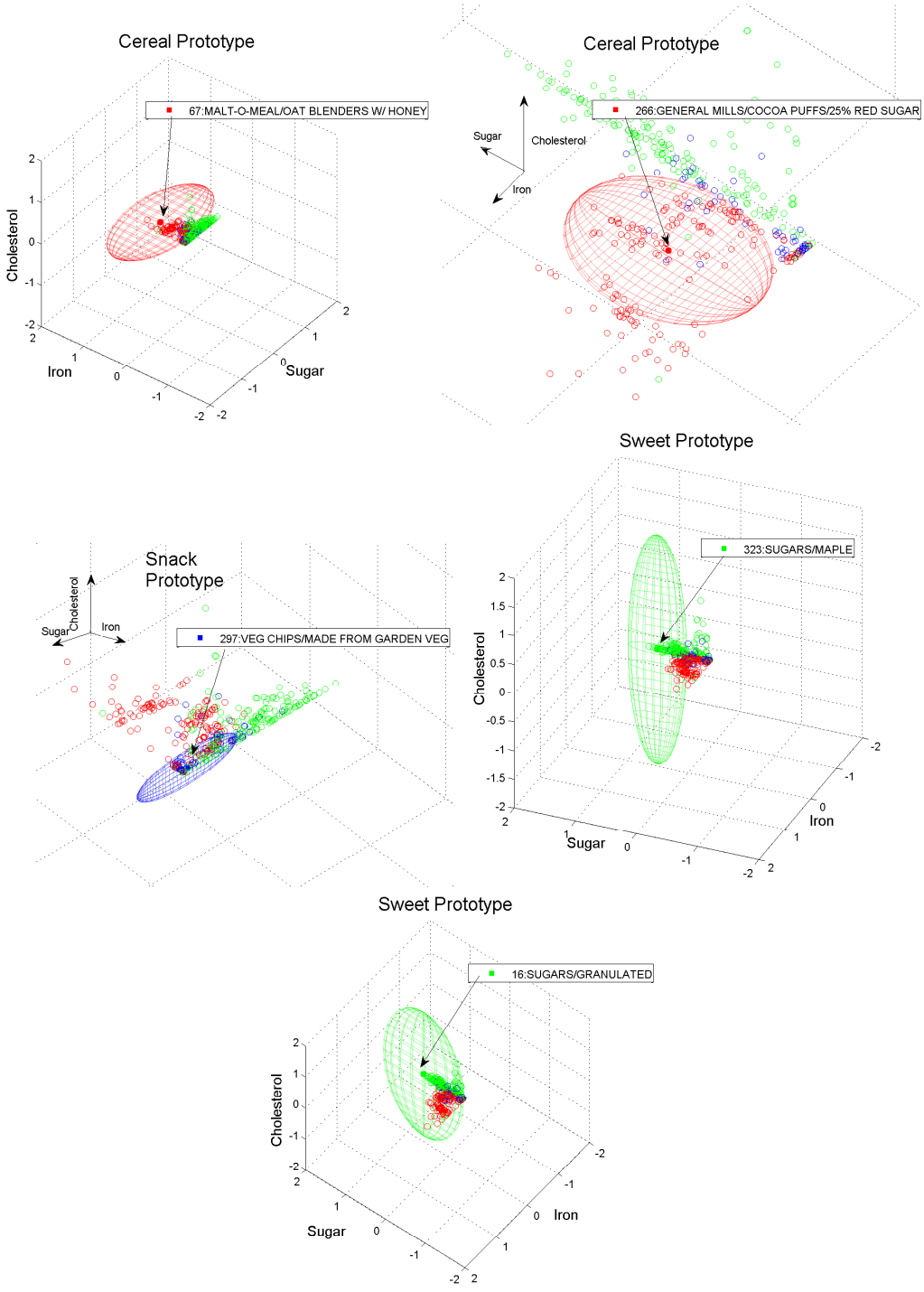


Figure 4: All of the five balls, in separate subfigures, with prototypes labeled.

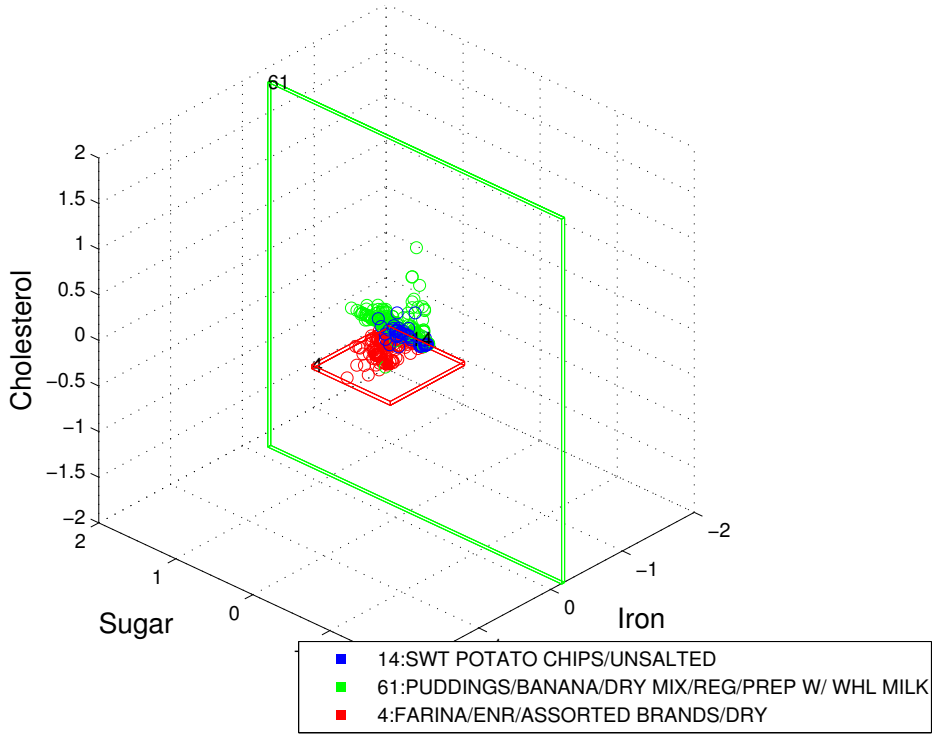


Figure 5: Hyper-rectangles for the nutrition data set. The method was initialized to have up to 30 rectangles and chose only three.

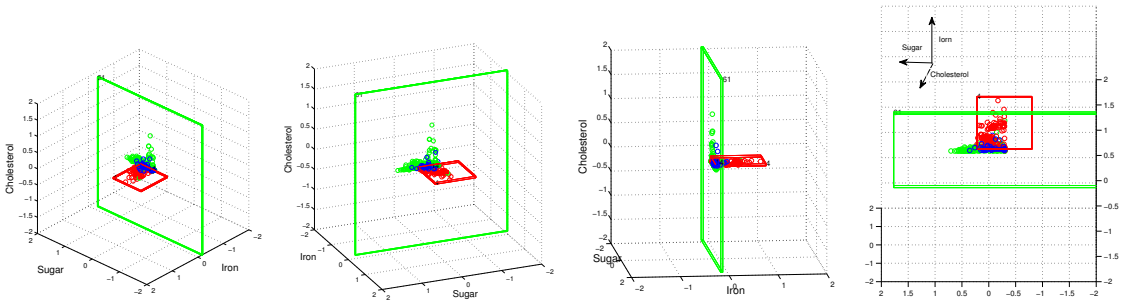


Figure 6: Rotated views of the hyper-rectangle classifier.

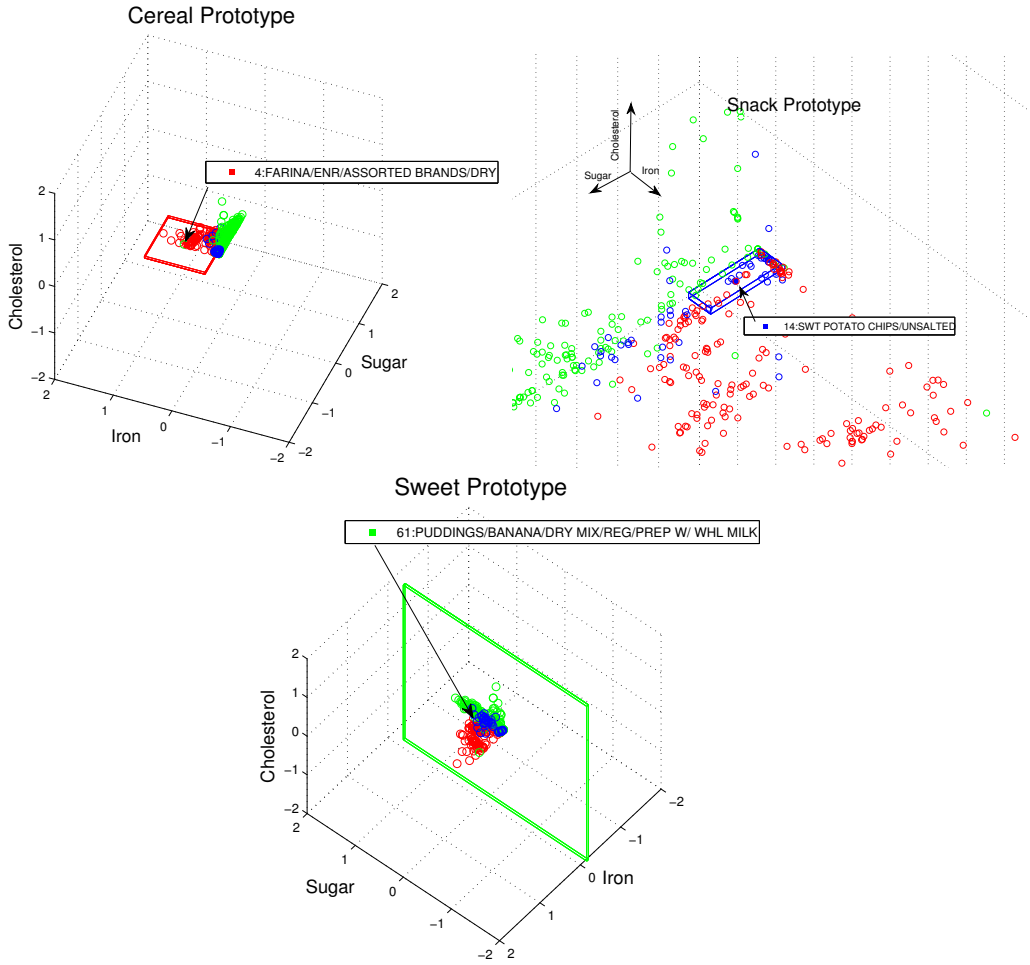


Figure 7: Views of the hyper rectangles for each class, with prototypes labeled in the figure.

of that class. The table also provides some examples of correctly classified food items from each of the balls.

Examples of correctly classified food items					
	Cereal 1	Cereal 2	Snack	Sweet 1	Sweet 2
Prototype Name	MALT-O-MEAL/ OAT BLENDERS W/ HONEY	GENERAL MILLS/COCOA PUFFS/25% RED SUGAR	VEG CHIPS/ MADE FROM GARDEN VEG	SUGARS/ MAPLE	SUGARS/ GRANU-LATED
Which ball	Large Red Ellipsoid	Small Red	Blue	Small Green	Large Green
Center location	Sugar: -0.074 Iron: 0.63 Chol:-0.032	Sugar:-0.034 Iron:0.121 Chol : -0.032	Sugar:-0.255 Iron:-0.102 Chol:-0.032	Sugar:0.554 Iron:-0.101 Chol:-0.032	Sugar:0.703 Iron:-0.127 Chol:-0.032
Axis lengths	Chol:0.03 Iron:0.72 Sugar:1.39	Chol:0.022 Iron:0.223 Sugar:0.288	Iron:0.015 Chol:0.117 Sugar:0.331	Iron:0.138 Sugar:0.48 Chol:2(infinite)	Iron:0.0309 Sugar:1.007 Chol:2(infinite)
Correct: incorrect ratio	164 : 22	106 : 9	32 : 8	101 : 10	135 : 7
Not covered	6		16	11	
Examples of Correctly Classified Food Items	• RTE/QUAKER/ KING VITAMAN Sugar: -0.093 Iron: 0.354 Chol: -0.032 • RTE/ GENERAL MILLS CINN CHEX Sugar: -0.034 Iron: 0.321 Chol: -0.032 • FIBER ONE BRAN CRL Sugar: -0.294 Iron: 0.121 Chol: -0.032	• KASHI GO LN HOT CRL CREAMY/ TRULY VANILLA/ DRY Sugar: -0.145 Iron: -0.070 Chol: -0.032 • KASHI ORGANIC PROMISE/ BERRY FRUITFUL Sugar: -0.149 Iron: -0.078 Chol: -0.032 • KELLOGG'S ALL-BRAN ORIGINAL Sugar: -0.139 Iron: 0.164 Chol: -0.032	• SWT POTATO CHIPS/ UNSALTED Sugar: -0.207 Iron: -0.093 Chol: -0.032 • PLAN-TAIN CHIPS/ SALTED Sugar: -0.287 Iron: -0.112 Chol: -0.032 • GRANOLA BARS/ SOFT/ COATD/ MILK CHOC COATING/ PNUT BUTTER Sugar: -0.058 Iron: -0.104 Chol: 0.053	• SWEET-ENER/SYRUP/ AGAVE Sugar: 0.385 Iron: -0.127 Chol: -0.032 • SUGARS/ GRANU-LATED Sugar: 0.703 Iron: -0.127 Chol: -0.032 • SYRUP/CANE Sugar: 0.437 Iron: -0.068 Chol: -0.032	• ICE CREAMS/ CHOC/LT Sugar: -0.048 Iron: -0.117 Chol: 0.167 • SWEET-ENER/ SYRUP/AGAVE Sugar:0.385 Iron:-0.127 Chol:-0.032 • ICE CREAMS/ BREYERS/ ALL NAT LT MINT CHOC CHIP Sugar:-0.041 Iron:-0.119 Chol:0.074

Table 1: The top part of this table provides the information about each prototype and the shape of its ball. Below that, we provide three more examples of correctly classified food items within each ball.

Examples of incorrectly classified food items					
	Cereal 1	Cereal 2	Snack	Sweet 1	Sweet 2
Prototype Name	MALT-O-MEAL/ OAT BLENDERS W/ HONEY	GENERAL MILLS/COCOA PUFFS/25% RED SUGAR	VEG CHIPS/ MADE FROM GARDEN VEG	SUGARS/ MAPLE	SUGARS/ GRANU-LATED
Which ball	Large Red Ellipsoid	Small Red	Blue	Small Green	Large Green
Center location	Sugar: -0.074 Iron:0.63 Chol:-0.032	Sugar:-0.034 Iron:0.121 Chol : -0.032	Sugar:-0.255 Iron:-0.102 Chol:-0.032	Sugar:0.554 Iron:-0.101 Chol:-0.032	Sugar:0.703 Iron:-0.127 Chol:-0.032
Axis lengths	Chol:0.03 Iron:0.72 Sugar:1.39	Chol:0.022 Iron:0.223 Sugar:0.288	Iron:0.015 Chol:0.117 Sugar:0.331	Iron:0.138 Sugar:0.48 Chol:2(infinite)	Iron:0.0309 Sugar:1.007 Chol:2(infinite)
Examples of food items that are incorrectly classified	• COCOA/ DRY PDR/ UNSWTND/ HERSHEY'S EUROPEAN STYLE COCOA ->(Sweets) Sugar:-0.296 Iron:0.471 Chol:-0.032 • FORMU-LATED BAR/ LUNA BAR/ NUTZ OVER CHOC ->(Snacks) Sugar:-0.126 Iron:0.108 Chol:-0.0326 • GRANO BAR/ GENER MILLS/ NATUR VALLE/ CHEWY TRAIL MIX ->(Snacks) Sugar:0.133 Iron:0.0166 Chol:-0.032	• FORMULATED/ BAR/ LUNA BAR/ NUTZ OVER CHOC ->(Snacks) Sugar:-0.126 Iron:0.108 Chol:-0.032 • CLIF BAR/ MIXED FLAVORS ->(Snacks) Sugar:0.0205 Iron:-0.0181 Chol:-0.032 • CANDIES/ SESAME CRUNCH ->(Sweets) Sugar:0.016 Iron:-0.057 Chol:-0.032	• GELATINS/ DRY PDR/ UNSWTND ->(Sweets) Sugar: -0.296 Iron: -0.110 Chol: -0.032 • PUDDINGS/ CHOC/ RTE ->(Sweets) Sugar: -0.124 Iron: -0.107 Chol: -0.0255 • CORN GRITS YEL REG & QUICK/ UNENR/ DRY ->(Cereals) Sugar:-0.289 Iron:-0.111 Chol:-0.032	• CANDY BITS/ YOGURT COVERED W/ VIT C ->(Snacks) Sugar: 0.355 Iron: -0.125 Chol: -0.011 • GRANOLA BAR/ FRUIT-FILLED/ NONFAT ->(Snacks) Sugar: 0.258 Iron: -0.061 Chol: -0.032 • PRETZEL/HARD/ CHOC COATD ->(Snacks) Sugar: 0.092 Iron: -0.086 Chol: -0.032	• GRAN BAR/ GENERAL MILLS/ NATURE VALLEY/ w YOGU COATI ->(Snacks) Sugar: 0.104 Iron: -0.111 Chol: -0.032 • FORMULATED BAR/ HIGH FIBER/ CHEWY/ OATS AND CHOCOLATE ->(Snacks) Sugar: -0.045 Iron: -0.113 Chol: -0.032 • BANANA CHIPS ->(Snacks) Sugar: 0.057 Iron: -0.107 Chol: -0.032

Table 2: Examples of food items that were misclassified by the ellipsoid method.

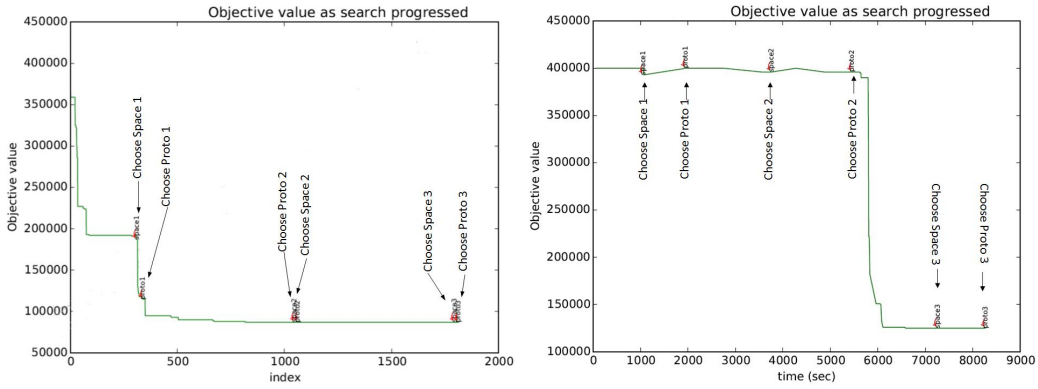


Figure 8: Example convergence traces. Objective function as time progresses, for the ellipsoidal alternating minimization method method (left), and the hyper-rectangles alternating minimization method (right). Convergence proceeds in jumps rather than gradually (which is common for discrete problems). There is a tag on the figures for each time a new optimization problem is started within the alternating minimization scheme.

Table 2 contains examples of food items that were misclassified. These are useful for gaining insight into the problem. In the first column, we list food items within the large red ellipsoid, which is one of the cereal ellipsoids. The red ellipsoid is characterized by low cholesterol levels and relatively high iron levels. The sugar content level is less important and varies widely in the ellipsoid. The first example is unsweetened cocoa powder, which was supposed to be classified under sweets. However, unsweetened cocoa has low sugar (unlike the sweets), low cholesterol similar to the cereals, and higher iron levels. Similarly the luna bar and nature valley granola bar are farther from the center of the cereal cluster, but still fall within the cluster because of their balance of low cholesterol, iron that is above the mean, and sugar levels that are not too far above the mean. Similarly, for the second column, the method misclassified several snacks as cereals because they fall into the ball's narrow cholesterol range and have midrange iron and sugar levels. In the third column, we see that gelatin, chocolate pudding, and corn grits were classified as snacks because their levels were all similar to the prototype, all below the mean, and in particular, iron levels within a narrow range. It is not clear why unsweetened gelatin was classified as a sweet in the first place. The first sweets ball is fairly wide, and tends to cover some of the snacks like the yogurt covered-candy bits with vitamins, and the chocolate covered pretzels. The second sweets ball has a very narrow low iron content, which is why the banana chips, high fiber bar, and granola bar fall into this ball.

Figure 8 shows the objective of the alternating minimization problem for the ellipsoid classifier (left), and for the hyper-rectangles (right) for sample runs. Each timepoint where the alternating formulation changed is tagged.

5.2 Comparison with other algorithms

We conducted a series of experiments to gauge the predictive accuracy and sparsity of the elastic prototype methods with respect to other similar machine learning methods, namely:

- C4.5 (Quinlan 1993) is one of the most popular algorithms in data mining. We compared with decision trees because they subdivide the space into axis-aligned rectangles. Decision trees are constructed in a greedy top-down way which gives them a disadvantage over our methods. C4.5 does not generally yield sparse trees unless heavily pruned. Decision trees do not use prototypes. We used the "J48" implementation in the R package RWeka. (Hornik et al. 2009, Witten & Frank 2005, Quinlan 1993).

- C5.0, Boosted decision trees. (Kuhn et al. 2012) is an updated decision tree algorithm that includes optional boosting. The "C5.0" function in the R package C50 was used (Kuhn et al. 2012). The number of boosting iterations was set to 10.

- CART (Breiman et al. 1984, Quinlan 1986) is a popular decision tree method that tends to produce very interpretable classifiers. We used the implementation of CART in the rpart R package.

- Protoclass (Bien & Tibshirani 2011) is a method that chooses prototypes, all with fixed sized balls. This method tends to choose a huge number of balls, because the balls are not allowed to stretch along the axes.

- k-medoids (Kaufman & Rousseeuw 1987) is a method that is similar to k-means except that each of the clusters has a point (a prototype) chosen from the original data set. Points are assigned to the nearest cluster center. We used the cluster's prototype to determine the predicted label of the cluster.

- pam-k (Hennig 2015) This method performs a partitioning around k-medoids clustering. The number of clusters or "k" is estimated by the algorithm using optimum average silhouette width. The silhouette width is defined by Rousseeuw (1987).

- Fast Boxes (Goh & Rudin 2014). This method is designed for imbalanced data sets. Its classifiers are axis-aligned rectangles that are optimized to surround the minority class. All points outside the rectangles are classified as majority class points. The method starts by clustering minority class points (k-means) and optimizing locally for the placement of the boxes around the clusters.

- Class cover catch digraphs, which is another prototype method that produces spherical balls. (Marchette 2015, 2004, Priebe et al. 2003).

Five-fold cross validation results are shown in Table 3, where one of the folds was the data set used for visualization in Section 5.1. The last three rows of Table 3 are the methods

proposed in this paper. Results for the ξ and η terms are each given in percentages, so the maximum percentage is 200%. For algorithms such as C4.5, C5.0, CART, and k-medoids that assign a single class to each point, note that $\xi = \eta$ since assigning a point to the wrong class ($\eta = 1$) is equivalent to failing to assign it to the correct class ($\xi = 1$). Protoclass assigns ξ and η separately during training, and during testing assigns test points to the nearest prototype, so for testing $\xi = \eta$. We trained k-medoids on each class separately to get prototypes specific to each class. We classified the test points according to their nearest prototype from the training set. The same procedure was followed for pam-k. We reported results for several different parameter values for Protoclass and k-medoids. We remark that the Fast Boxes algorithm was meant to be used only for heavily imbalanced data sets, and our data set is not heavily imbalanced. Its performance degrades because of this. In our implementation of Fast Boxes, we used a separate box drawing classifier for each class label, and combined them to create hyper-rectangles (but without a prototype for each hyper-rectangle).

The important columns in Table 3 are the test error "combined" column and the number of prototypes column. These two columns convey the error rate and sparsity of the result. The main observations are:

1. The prototype methods (k-medoids, class cover catch digraphs, and Protoclass) tended not to be particularly accurate for this task. This holds for a variety of different parameter settings. The pam-k algorithm performed slightly better, especially given its small number of prototypes, but did not get to the accuracy of the ellipsoid or hyper-rectangle alternating minimization schemes.

2. The number of prototypes for Protoclass tended to be much larger (often above 100) than for the elastic prototype methods (less than 10). A similar observation holds for class cover catch digraphs, which overfit the training set.

3. The accuracy of the ellipsoids and hyper-rectangles was approximately at the same level as (or slightly worse than) the performance level of the decision trees. Decision trees are not directly comparable to the methods proposed here in ways other than accuracy, as they do not use prototypes and there is not a notion of being classified into multiple labels simultaneously. They are also not customizable.

To summarize, the elastic prototype methods proposed here have advantages over other methods, namely (i) they have a good balance of accuracy and sparsity, as discussed just above, (ii) they use mixed integer linear programming, which is methodologically principled and allows for customization, (iii) they can be useful for data understanding and visualization.

One of the main benefits, that has not been addressed so far, is the ability to be flexible, meaning that we can easily add extra user-defined constraints to the mixed-integer programming formulations and produce highly customized models. For instance, we could

customize the size and shapes of the balls to suit our interests. These types of constraints could not be added to greedy methods like decision trees. Trying to add constraints into greedy splitting and pruning criteria would be almost impossible in practice.

Our methods do make sacrifices for these benefits. In particular, when data set sizes become too large, this type of work does not make sense, whereas a decision tree, constructed in a top-down greedy way, would still be sensible. (Then accordingly, one loses the nice features of the elastic prototypes.)

Table 3: Comparison of Methods

Algorithm	Parameters	Train			Test			# of Prototypes
		ξ	η	combined	ξ	η	combined	
C4.5	default	10.17± (1.69)	10.17± (1.69)	20.34± (3.38)	13.2± (3.83)	13.2± (3.83)	26.4± (7.66)	#leaves :14.6±(2.5) tree size :28.2±(5.01)
C5.0	default	8.21± (1.07)	8.21± (1.07)	16.42± (2.14)	15.6 ± (3.78)	15.6 ± (3.78)	31.2± (7.56)	#leaves : 12.46± (2.71)
CART	default	14.12± (0.83)	14.12± (0.83)	28.24± (1.66)	17.8± (6.68)	17.8± (6.68)	35.6± (13.36)	#nodes : 17.8 ± (10.7)
Protoclass	eps=0.1	53.5± (2.75)	13.15± (3.86)	66.65± (3.17)	58.8± (5.31)	58.8± (5.31)	117.6± (10.62) $\xi = \eta$	class1: 33.3±(16.6) class2: 30.8±(15.2) class3: 7.83±(4.49)
	eps=0.01	22.05± (2.65)	1.95± (0.693)	24.0± (2.69)	62.6± (1.67)	62.6± (1.67)	124.4± (3.34) $\xi = \eta$	class1:110.6±(54.7) class2: 99.8±(49.0) class3: 35.8±(17.6)
	eps=0.001	15.95± (0.693)	0.95± (0.325)	16.9± (0.99)	62.2± (3.63)	62.2± (3.63)	124.4± (7.26) $\xi = \eta$	class1:122.16±(59.9) class2:109.8±(53.9) class3: 40.5±(19.9)
K-medoids	k=2 per class for 3 classes	49.7± (9.29)	49.7± (9.29)	99.4± (18.58)	47.6± (14.31)	47.6± (14.31)	95.2± (28.62)	medoids : 6
	k=3 per class for 3 classes	33.4± (10.86)	33.4± (10.86)	66.6± (21.6)	33.8± (11.58)	33.8± (11.58)	67.6± (23.16)	medoids : 9
	k=4 per class for 3 classes	32.55± (8.06)	32.55± (8.06)	65.1± (16.12)	35.6± (12.03)	35.6± (12.03)	71.2± (24.06)	medoids : 12
	k=58 per class for 3 classes	13.75± (1.36)	13.75± (1.36)	27.5± (2.72)	21.8± (7.59)	21.8± (7.59)	43.6± (15.18)	medoids : 174
pam-k	default	42.6± (15.3)	42.6± (15.3)	85.2± (30.6)	42.2± (16.14)	42.2± (16.14)	84.4± (32.2)	medoids : 7±(2.17)
Class Cover Catch Digraph	default	0.0	0.0	0.0	19.4± (5.3)	19.4± (5.3)	38.8± (10.6)	94± (7.51)
Fastboxes	csize=10 boxes=10 beta=1	14.53± (0.13)	14.53± (0.13)	29.06± (0.26)	23.33± (0.11)	23.33± (0.11)	46.66± (0.22)	10
Rectangles, hierarchical	$C_k = 1000$ $C_n = 1000$ $C_\alpha = .0001$ $C_\beta = .0001$ $C_\omega = .0001/w_2$ $C_v = 0.0001$	10± (0.9)	25± (9.83)	35± (9.62)	24.4± (5.94)	16.2± (5.02)	40.6± (6.58)	5
Ellipse alternating	3 iterations	13.35± (2.78)	12.10± (4.55)	25.45± (4.71)	17.20± (3.42)	14.40± (6.73)	31.65± (5.77)	5
Rectangular alternating	3 iterations	24.95± (5.11)	11.25± (4.37)	36.20± (8.67)	24.20± (11.48)	17.20± (8.93)	41.40± (9.18)	4.5± (1)

6. Conclusion

As a fundamental mode of reasoning, humans compare present situations to past cases. In this work we presented a classification method that identifies important prototypes from each class. Each prototype is surrounded by an elastic ball of similar observations, which in this work is either an ellipsoid or hyper-rectangle. Because the balls are elastic, observations can to be compared to each other along the important dimensions for that particular comparison. These elastic balls can help gain insight into the data and be visualized to help with data understanding.

There are several opportunities for building on this work. One option is to allow for non-axis aligned ellipsoids. Our current formulation can be straightforwardly generalized to this case by removing the requirement that $\Omega^{(b)}$ be diagonal. In this case, the constraints on the ellipsoids' principal axes' lengths become eigenvalue constraints, meaning that the mixed-integer linear program becomes a much more difficult to solve mixed-integer semidefinite program. We chose not to pursue this direction because non-axis aligned ellipses would not be easily interpretable to users. Another option for future work is to aim to improve the hierarchical warm start method, which did not seem to yield performance gains similar to the alternating minimization scheme. A third possibility for future study is to reformulate or specialize the optimization problems for specialized problems. A fourth possibility is to design more heuristic schemes that will scale to larger sizes, at the possible expense of lower accuracy; for these schemes, the present work provides a useful and principled basis for comparison. We showed that the previous methods for heuristically solving this problem have serious flaws, in that they tend to produce an excessively large numbers of prototypes or balls. It may be possible to improve those methods to produce better results.

Elastic Prototype Classifiers build off a foundation in psychology, mathematical programming, set covering, and supervised machine learning. Because they use mathematical programming, they can be substantially more useful, flexible, and customizable for data exploration than other types of methods. Adding structure in the form of user-defined constraints can help to offset the extra "elasticity" one might choose to add to the balls through adjusting the regularization constants. More generally, if desired, users could trade off between their own prior knowledge, the number of balls, elasticity of the balls, the number of prototypes and size of the subspace. This flexibility might allow users to optimize not only for the quality of the predictions but also for the reasons behind them.

7. Code Supplement

Our code is available as a supplement to this manuscript at
`https://github.com/prashan/elastic_prototype`

acknowledgements

Funding for this project was provided in part by Ford Motors, to C. Rudin.

References

Aamodt, A. (1991), 'A knowledge-intensive, integrated approach to problem solving and sustained learning', *Knowledge Engineering and Image Processing Group. University of Trondheim* pp. 27–85.

Bereg, S., Cabello, S., Díaz-Báñez, J. M., Pérez-Lantero, P., Seara, C. & Ventura, I. (2012), 'The class cover problem with boxes', *Computational Geometry* **45**(7), 294–304.

Bien, J. & Tibshirani, R. (2011), 'Prototype selection for interpretable classification', *Annals of Applied Statistics* **5**(4), 2403–2424.

Biswas, S., Sinha, N. & Purkayastha, B. (2014), 'A review on fundamentals of case-based reasoning and its recent application in different domains', *International Journal of Advanced Intelligence Paradigms* **6**(3), 235–254.

Bixby, R. E. (2010), 'Mixed-integer programming: It works better than you may think, presentation slides', http://www.ferc.gov/CalendarFiles/20100609110044-Bixby,%20Gurobi%20Optimization.pdf. accessed: Jan 2, 2016.

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984), *Classification and Regression Trees*, Wadsworth.

Carroll, J. (1980), Analyzing decision behavior: The magician's audience, *in* 'Cognitive processes in choice and decision behavior', Lawrence Erlbaum Associates, Inc. Hillsdale, NJ.

Cohen, M., Freeman, J. & Wolf, S. (1996), 'Metarecognition in time-stressed decision making: Recognizing, critiquing, and correcting', *Human Factors* **38(2)**, 206–219.

Cunningham, P., Doyle, D. & Loughrey, J. (2003), An evaluation of the usefulness of case-based explanation, *in* 'CBRRD', Springer.

Efrat, A., Hoffmann, F., Knauer, C., Kriegel, K., Rote, G. & Wenk, C. (2004), 'Covering with ellipses', *Algorithmica* **38**(1), 145–160.

Friedman, J. H. & Fisher, N. I. (1999), 'Bump hunting in high-dimensional data', *Statistics and Computing* **9**(2), 123–143.

Garg, V. K., Rudin, C. & Jaakkola, T. S. (2015), 'CRAFT: cluster-specific assorted feature selection', *CoRR* **abs/1506.07609**.
URL: *http://arxiv.org/abs/1506.07609*

Goh, S. & Rudin, C. (2014), Box drawings for learning with imbalanced data, *in* 'KDD'.

Guan, Y., Dy, J. G. & Jordan, M. I. (2011), A unified probabilistic model for global and local unsupervised feature selection, *in* 'ICML'.

Hennig, C. (2015), 'pamk function in fpc rpackage', http://www.inside-r.org/packages/cran/fpc/docs/pamk.

Hornik, K., Buchta, C. & Zeileis, A. (2009), 'Open-source machine learning: R meets Weka', *Computational Statistics* **24**(2), 225–232.

Huggins, J. H. & Rudin, C. (2014), A statistical learning theory framework for supervised pattern discovery, *in* 'In Proceedings of SIAM Conference on Data Mining (SDM)'.

Kaufman, L. & Rousseeuw, P. (1987), Clustering by means of medoids, *in* Y. Dodge, ed., 'Statistical Data Analysis Based on the L_1 Norm and Related Methods', North-Holland, pp. 405–416.

Kaufman, L. & Rousseeuw, P. J. (1990), 'Partitioning around medoids (program pam)', *Finding groups in data: an introduction to cluster analysis* pp. 68–125.

Kim, B., Rudin, C. & Shah, J. (2014), The bayesian case model: A generative approach for case-based reasoning and prototype classification, *in* 'Proceedings of Neural Information Processing Systems (NIPS)'.

Klein, G. (1989), 'Do decision biases explain too much', *Human Factors Society Bulletin* **23**(5), 1–3.

Kuhn, M., Weston, S. & Coulter, N. (2012), *C50: C5.0 Decision Trees and Rule-Based Models, C code for C5.0 by R. Quinlan.* R package version 0.1.0-013.
URL: *http://CRAN.R-project.org/package=C50*

Lazzeroni, L. & Owen, A. (2002), 'Plaid models for gene expression data', *Statistica Sinica* **12**, 61–86.

Madrigal, S. & Knopp, K. (2008), *Superflex: A Superhero Social Thinking Curriculum Package*, Think Social Publishing, Inc.

Malioutov, D. & Varshney, K. (2013), Exact rule learning via boolean compressed sensing, *in* 'ICML'.

Marchand, M. & Taylor, J. S. (2003), 'The set covering machine', *The Journal of Machine Learning Research* **3**, 723–746.

Marchette, D. J. (2004), 'Class cover catch digraphs', *Random Graphs for Statistical Pattern Recognition* pp. 129–184.

Marchette, D. J. (2015), 'Class cover catch digraphs, package cccd in rpackage', https://cran.r-project.org/web/packages/cccd/cccd.pdf.

Newell, A. & Simon, H. (1972), *Human problem solving*, Prentice-Hall Englewood Cliffs.

Priebe, C. E., Marchette, D. J., DeVinney, J. G. & Socolinsky, D. A. (2003), 'Classification using class cover catch digraphs', *Journal of classification* **20**(1), 003–023.

Quinlan, J. R. (1986), 'Induction of decision trees', *Machine learning* **1**(1), 81–106.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Rosch, E. H. (1973), 'Natural categories', *Cognitive Psychology* **4**, 328–350.

Rousseeuw, P. J. (1987), 'Silhouettes: A graphical aid to the interpretation and validation of cluster analysis', *Journal of Computational and Applied Mathematics* **20**, 53 – 65.

Slade, S. (1991), 'Case-based reasoning: A research paradigm', *AI magazine* **12(1)**, 42–55.

Tipping, M. & Schölkopf, B. (2001), 'A kernel approach for vector quantization with guaranteed distortion bounds', *Artificial Intelligence and Statistics* pp. 129–134.

U.S. Department of Agriculture, Agricultural Research Service (2013), 'USDA national nutrient database for standard reference, release 26. nutrient data laboratory home page', http://www.ars.usda.gov/ba/bhnrc/ndl.

Wang, T., Rudin, C., Wagner, D. & Sevieri, R. (2013), Detecting patterns of crime with series finder, *in* 'Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2013)'.

Wang, T., Rudin, C., Wagner, D. & Sevieri, R. (2015), 'Finding patterns with a rotten core: Data mining for crime series with core sets', *Big Data* **3**.

Witten, I. H. & Frank, E. (2005), *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edn, Morgan Kaufmann, San Francisco.