Stochastic Optimization of Multiple Objectives and Supervised Machine Learning

Luis Nunes Vicente

ISE, Lehigh University

Dept. of Industrial Engineering, Univ. of Pittsburgh

November 14, 2019

Optimization models in Data Science and Learning

A data set for analysis involving optimization is typically of the form

$$D = \{(a_j, y_j), j = 1, \ldots, N\}$$

where the a_j's vectors are features or attributes

the y_j's vectors are labels or observation or responses.

Optimization models in Data Science and Learning

A data set for analysis involving optimization is typically of the form

$$D = \{(a_j, y_j), j = 1, \dots, N\}$$

where the a_j's vectors are features or attributes

the y_j's vectors are labels or observation or responses.

The analysis consists of finding a prediction function $\phi(a_j)$ such that

$$\phi(a_j) \simeq y_j, \quad j = 1, \dots, N$$

in some optimal sense.

Optimization models in Data Science and Learning

A data set for analysis involving optimization is typically of the form

$$D = \{(a_j, y_j), j = 1, \ldots, N\}$$

where the a_j's vectors are features or attributes

the y_j's vectors are labels or observation or responses.

The analysis consists of finding a prediction function $\phi(a_j)$ such that

$$\phi(a_j) \simeq y_j, \quad j = 1, \ldots, N$$

in some optimal sense.

Often, ϕ is parameterized $\phi(x) = \phi(a; x)$. The parameters are x.

Notes:

1. The process of finding ϕ is called learning or training.

Notes:

1. The process of finding ϕ is called learning or training.

2. When the y_j's are reals, one has a regression problem.

Notes:

1. The process of finding ϕ is called learning or training.
2. When the y_j's are reals, one has a regression problem.
3. When the y_j's lie in a finite set $\{1, \ldots, M\}$, one has a classification problem.

   $M = 2$ leads to binary classification.

Notes:

1. The process of finding ϕ is called learning or training.

2. When the y_j's are reals, one has a regression problem.

3. When the y_j's lie in a finite set $\{1, \ldots, M\}$, one has a classification problem.

   $M = 2$ leads to binary classification.

4. The labels may be null. In that case, one may want:

   to group the a_j's in clusters (clusterization)

   or to identify a low-dimensional subspace (or a collection of) where the a_j's lie (subspace identification).

Notes:

1. The process of finding ϕ is called learning or training.

2. When the y_j's are reals, one has a regression problem.

3. When the y_j's lie in a finite set $\{1, \ldots, M\}$, one has a classification problem.

   $M = 2$ leads to binary classification.

4. The labels may be null. In that case, one may want:

   to group the a_j's in clusters (clusterization)

   or to identify a low-dimensional subspace (or a collection of) where the a_j's lie (subspace identification).

   The labels may have to be learned while learning ϕ.

5. Data is assumed to be clean for optimization, but still:

   i) (a_j, y_j)'s could be noisy or corrupted.
   ii) Some a_j or y_j's could be missing.
   iii) Data could arrive in streaming fashion (ϕ must be learned online).

5. Data is assumed to be clean for optimization, but still:

   i) (a_j, y_j)'s could be noisy or corrupted.
   ii) Some a_j or y_j's could be missing.
   iii) Data could arrive in streaming fashion (ϕ must be learned online).

   Thus, ϕ has to be robust to changes in the data set.

5. Data is assumed to be clean for optimization, but still:

   i) (a_j, y_j)'s could be noisy or corrupted.
   ii) Some a_j or y_j's could be missing.
   iii) Data could arrive in streaming fashion (ϕ must be learned online).

   Thus, ϕ has to be robust to changes in the data set.

Such data analysis is often referred to as machine learning or data mining.

$$\begin{tikzcd}
 & \text{machine learning} \arrow[dl] \arrow[dr] & \text{data mining} \arrow[d] \\
\text{predictive or supervised learning (when labels exist)} & & \text{unsupervised learning (when labels are null): extract interesting information from data}
\end{tikzcd}$$

The correct/accurate match of the data is typically quantified by a loss function $\ell(a, y; \phi(x))$ and thus learning can be formulated as

$$\min_x \sum_{j=1}^{N} \ell(a_j, y_j; \phi(x))$$

The correct/accurate match of the data is typically quantified by a loss function $\ell(a, y; \phi(x))$ and thus learning can be formulated as

$$\min_x \sum_{j=1}^{N} \ell(a_j, y_j; \phi(x))$$

To avoid overfitting such a model to a sample D, one often adds a regularizer

$$\begin{tikzcd} & \displaystyle\min_x \sum_{j=1}^{N} \ell(a_j, y_j; \phi(x)) + g(x) \arrow[dl] \arrow[dr] & \\ \lambda\|x\|_1 & & \frac{\lambda}{2}\|x\|_2^2 \end{tikzcd}$$

so that ϕ is not so sensitive to changes in D.

The general form of optimization models in DS is

$$\min_{x \in \mathbb{R}^n} \; f(x) + g(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is smooth.

The general form of optimization models in DS is

$$\min_{x \in \mathbb{R}^n} \; f(x) + g(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is smooth.

∇f is at least Lipschitz continuous.

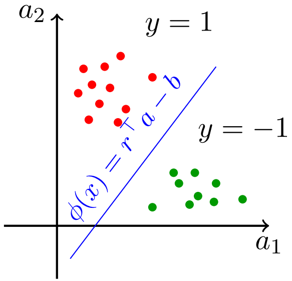
$g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is convex, proper and closed.

g is typically non-smooth.

Classical example: logistic regression for binary classification

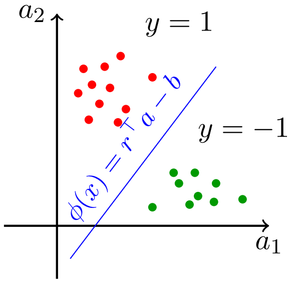
Classical example: logistic regression for binary classification

Consider a data set with $y \in \{-1, 1\}$



Classical example: logistic regression for binary classification

Consider a data set with $y \in \{-1, 1\}$



$\phi(x) = \phi(a; r, b) = r^\top a - b$ is a linear classifier. One aims to seek for a optimal parameter $x = (r, b)$ such that

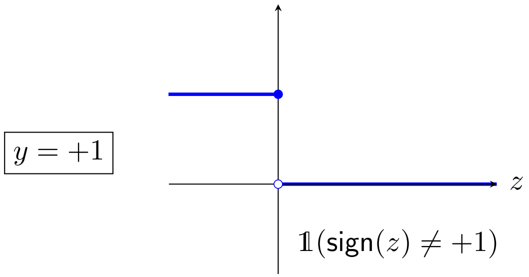
$$\begin{cases} r^\top a_j - b \ \geq \ 0 & \text{when } y_j = 1 \\ r^\top a_j - b \ < \ 0 & \text{when } y_j = -1 \end{cases} \qquad \forall j = 1, \dots, N \quad (*)$$

Logistic regression can be motivated by two arguments.

1. Smoothing/convexifying the true loss.

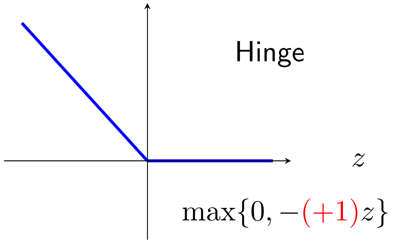
   What we want to do is

$$\min \ \mathbb{1}(\mathsf{sign}(z) \neq y) \qquad \text{where } z = \phi(x).$$



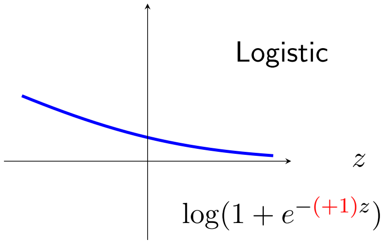
Smoothing/convexifying to Hinge loss (SVM)

$\boxed{y = +1}$



Smoothing to Logistic loss

$\boxed{y = +1}$

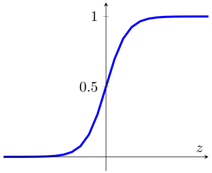


2. Probabilistic argument: $(*)$ is equivalent to

$$\sigma(\phi(x)) \;=\; \frac{1}{1+e^{-\phi(x)}} \begin{cases} \geq 0.5 & \text{when } y = 1 \\ < 0.5 & \text{when } y = -1 \end{cases}$$

2. Probabilistic argument: $(*)$ is equivalent to

$$\sigma(\phi(x)) = \frac{1}{1+e^{-\phi(x)}} \begin{cases} \geq 0.5 & \text{when } y = 1 \\ < 0.5 & \text{when } y = -1 \end{cases}$$



$\frac{1}{1+e^{-z}}$

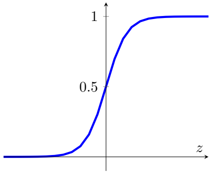
Think of it as the probability

$$P(y=1|a) = \sigma(\phi(x)) = \frac{1}{1+e^{-(+1)\phi(x)}}$$

$$P(y=-1|a) = 1-\sigma(\phi(x)) = \frac{1}{1+e^{-(-1)\phi(x)}}$$

2. Probabilistic argument: $(*)$ is equivalent to

$$\sigma(\phi(x)) = \frac{1}{1+e^{-\phi(x)}} \begin{cases} \geq 0.5 & \text{when } y = 1 \\ < 0.5 & \text{when } y = -1 \end{cases}$$



$\frac{1}{1+e^{-z}}$

Think of it as the probability

$$P(y=1|a) = \sigma(\phi(x)) = \frac{1}{1+e^{-(+1)\phi(x)}}$$

$$P(y=-1|a) = 1-\sigma(\phi(x)) = \frac{1}{1+e^{-(-1)\phi(x)}}$$

Assuming data points are i.i.d., the maximum likelihood problem is

$$\max \prod_{j=1}^{N} \underbrace{P(y_j|a_j)}_{\text{prob. of correct prediction}} = \prod_{j=1}^{N} \frac{1}{1+e^{-y_j\phi(x)}}$$

Then, the negative log likelihood leads to the loss minimization problem

$$\min_{r,b} \; \frac{1}{N} \sum_{j=1}^{N} \ell_L(a_j, y_j; \phi(x))$$

with the smooth convex logistic loss function

$$\ell_L(a_j, y_j; \phi(x)) \; = \; \log(1 + e^{-y_j(r^\top a_j - b)})$$

Then, the negative log likelihood leads to the loss minimization problem

$$\min_{r,b} \frac{1}{N} \sum_{j=1}^{N} \ell_L(a_j, y_j; \phi(x))$$

with the smooth convex logistic loss function

$$\ell_L(a_j, y_j; \phi(x)) \;=\; \log(1 + e^{-y_j(r^\top a_j - b)})$$

Binary classification is formulated as the optimization problem

$$\min_{r,b} \frac{1}{N} \sum_{j=1}^{N} \ell_L(a_j, y_j; \phi(x)) + \frac{\lambda}{2}\|r\|_2^2$$

Presentation outline

Stochastic gradient descent for Stochastic Optimization

Assume a point $(a, y) \in D$ arises with a certain (joint) probability P_Ω.

Stochastic gradient descent for Stochastic Optimization

Assume a point $(a, y) \in D$ arises with a certain (joint) probability P_Ω.

The general goal now is to minimize the expected risk of misclassification

$$R(x) \; = \; \int \ell(a, y; \phi(x)) dP_\Omega(a, y)$$

Stochastic gradient descent for Stochastic Optimization

Assume a point $(a, y) \in D$ arises with a certain (joint) probability P_Ω.

The general goal now is to minimize the expected risk of misclassification

$$R(x) \; = \; \int \ell(a, y; \phi(x)) dP_\Omega(a, y)$$

When P_Ω is unknown, only having a sample D leads to minimizing the empirical risk of misclassification

$$R_N(x) \; = \; \frac{1}{N} \sum_{j=1}^{N} \ell(a_j, y_j; \phi(x))$$

as an estimation of $R(x)$.

For simplicity, let $f(x) = \ell(a, y; \phi(x))$.

For simplicity, let $f(x) = \ell(a, y; \phi(x))$.

w is the random seed variable representing a sample. (A set of realizations $\{w_{[j]}\}_{j=1}^{N}$ corresponds to a sample set $\{(a_j, y_j), j = 1, \ldots, N\}$.)

For simplicity, let $f(x) = \ell(a, y; \phi(x))$.

w is the random seed variable representing a sample. (A set of realizations $\{w_{[j]}\}_{j=1}^{N}$ corresponds to a sample set $\{(a_j, y_j), j = 1, \ldots, N\}$.)

The general objective could be written as

$$R(x) = \begin{cases} R(x) = \mathbb{E}[f(x; w)] & \text{Online setting (expected risk)} \\ R_N(x) = \frac{1}{N} \sum_{j=1}^{N} f_j(x) & \text{Finite sum setting (empirical risk)} \end{cases}$$

where $f_j(x) = f(x; w_{[j]})$ is the loss associated with the j-th sample.

For simplicity, let $f(x) = \ell(a, y; \phi(x))$.

w is the random seed variable representing a sample. (A set of realizations $\{w_{[j]}\}_{j=1}^{N}$ corresponds to a sample set $\{(a_j, y_j), j = 1, \ldots, N\}$.)

The general objective could be written as

$$R(x) = \begin{cases} R(x) = \mathbb{E}[f(x; w)] & \text{Online setting (expected risk)} \\ R_N(x) = \frac{1}{N} \sum_{j=1}^{N} f_j(x) & \text{Finite sum setting (empirical risk)} \end{cases}$$

where $f_j(x) = f(x; w_{[j]})$ is the loss associated with the j-th sample.

Both can be minimized by the stochastic gradient method.

The stochastic gradient method

Algorithm 1 Stochastic Gradient (SG) method (Robbins and Monro 1951)

```
1: Choose an initial point $x_0 \in \mathbb{R}^n$.
2: for $k = 0, 1, \dots$ do
3:     Compute a stochastic gradient $g(x_k; w_k)$.
4:     Choose a step-size $\alpha_k > 0$.
5:     Set the new iterate $x_{k+1} = x_k - \alpha_k g(x_k; w_k)$.
6: end for
```

Sutton Monro
Lehigh ISE

The stochastic gradient method

Algorithm 1 Stochastic Gradient (SG) method (Robbins and Monro 1951)

```
1: Choose an initial point $x_0 \in \mathbb{R}^n$.
2: for $k = 0, 1, \ldots$ do
3:     Compute a stochastic gradient $g(x_k; w_k)$.
4:     Choose a step-size $\alpha_k > 0$.
5:     Set the new iterate $x_{k+1} = x_k - \alpha_k g(x_k; w_k)$.
6: end for
```

Sutton Monro
Lehigh ISE

The stochastic gradient $g(x_k; w_k)$ could be

$$\nabla f(x_k; w_k) \;=\; \nabla f_{j_k}(x_k) \qquad \text{simple or basic SG}$$

The stochastic gradient method

Algorithm 1 Stochastic Gradient (SG) method (Robbins and Monro 1951)

```
1: Choose an initial point $x_0 \in \mathbb{R}^n$.
2: for $k = 0, 1, \ldots$ do
3:     Compute a stochastic gradient $g(x_k; w_k)$.
4:     Choose a step-size $\alpha_k > 0$.
5:     Set the new iterate $x_{k+1} = x_k - \alpha_k g(x_k; w_k)$.
6: end for
```

Sutton Monro
Lehigh ISE

The stochastic gradient $g(x_k; w_k)$ could be

$$\nabla f(x_k; w_k) \;=\; \nabla f_{j_k}(x_k)$$ simple or basic SG

$$\frac{1}{|B_k|} \sum_{j \in B_k} \nabla f(x_k; w_{k,j})$$ mini-batch SG ($|B_k|$ is the mini-batch size)

Classical assumptions for SG method

1. The sequence of realizations $\{w_k\}$ are i.i.d. sample of w.
2. Function R has Lipschitz continuous gradient.
3. The stochastic gradient is an unbiased estimate

$$\mathbb{E}_{w_k}[g(x_k; w_k)] \;=\; \nabla R(x_k)$$

4. The stochastic gradient has bounded variance

$$\mathbb{V}_{w_k}[g(x_k; w_k)] \;\leq\; M + \mathcal{O}(\|\nabla R(x_k)\|^2)$$

5. Function R is bounded below: $R(x) \geq R_*, \forall x$. $R_* = R(x_*)$ where x_* is a minimizer of R.

$\mathbb{E}[\cdot]$ denotes the expected value taken w.r.t. the joint distribution of $\{w_k\}$.

Convergence rates for SG method

The strongly convex case (Sacks 1958)

Consider a diminishing step-size sequence (e.g., $\alpha_k \simeq \mathcal{O}(1/k)$), one has

$$\mathbb{E}[R(x_k)] - R_* \leq \mathcal{O}(1/k)$$

Convergence rates for SG method

The strongly convex case (Sacks 1958)

Consider a diminishing step-size sequence (e.g., $\alpha_k \simeq \mathcal{O}(1/k)$), one has

$$\mathbb{E}[R(x_k)] - R_* \;\leq\; \mathcal{O}(1/k)$$

The convex case (Nemirovski and Yudin 1978)

Consider a diminishing step-size sequence (e.g., $\alpha_k \simeq \mathcal{O}(1/\sqrt{k})$), one has

$$\mathbb{E}[R^k_{\text{best}}] - R_* \;\leq\; \mathcal{O}(1/\sqrt{k})$$

where $R^k_{\text{best}} = \min_{1\leq i\leq k} R(x_i)$.

Presentation outline

Multi-Objective Optimization

Multi-Objective Optimization (MOO) deals with multiple potentially conflicting objectives simultaneously.

Multi-Objective Optimization

Multi-Objective Optimization (MOO) deals with multiple potentially conflicting objectives simultaneously.

We consider smooth MOO problems of the general form

$$\begin{aligned} \min \;\; & H(x) \; = \; (h_1(x), \ldots, h_m(x)) \\ \text{s.t.} \;\; & x \; \in \; \mathcal{X} \end{aligned}$$

where $h_i : \mathbb{R}^n \to \mathbb{R}$ are smooth functions and $\mathcal{X}$ is a feasible region.

Multi-Objective Optimization

Multi-Objective Optimization (MOO) deals with multiple potentially conflicting objectives simultaneously.

We consider smooth MOO problems of the general form

$$\begin{aligned} \min \ & H(x) = (h_1(x), \ldots, h_m(x)) \\ \text{s.t. } & x \in \mathcal{X} \end{aligned}$$

where $h_i : \mathbb{R}^n \to \mathbb{R}$ are smooth functions and $\mathcal{X}$ is a feasible region.

Definition of optimality for MOO

- x dominates y if $H(x) < H(y)$ componentwise, $\forall x, y \in \mathcal{X}$.
- x is a Pareto minimizer if it is not dominated by any other point in $\mathcal{X}$.

  Also called a non-dominated or efficient point.

Pareto front

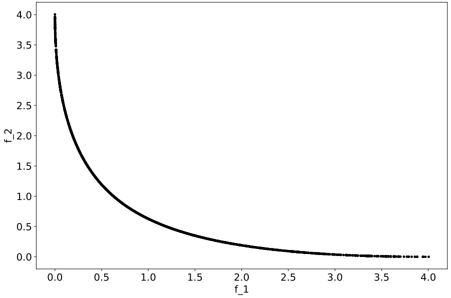


Figure: Pareto front of problem SP1

Denote $\mathcal{P}$ as the set of Pareto minimizers.

Pareto front is a mapping from $\mathcal{P}$ to function value space $\mathbb{R}^m$, i.e.,

$$H(\mathcal{P}) = \{H(x) : x \in \mathcal{P}\}$$

Pareto front

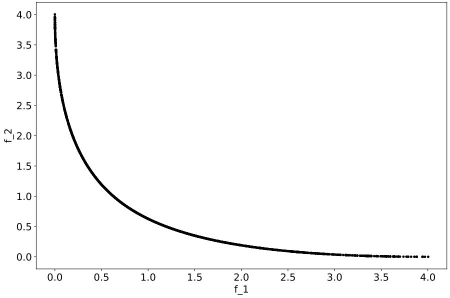


Figure: Pareto front of problem SP1

Denote $\mathcal{P}$ as the set of Pareto minimizers.

Pareto front is a mapping from $\mathcal{P}$ to function value space $\mathbb{R}^m$, i.e.,

$$H(\mathcal{P}) = \{H(x) : x \in \mathcal{P}\}$$

The goal of MOO: find the set of Pareto minimizers and hence Pareto front to define the best trade-off among several competing criteria.

Necessary condition for Pareto minimizers

Pareto first-order stationary condition: x is a Pareto stationary point if

$$\nexists\, d \in \mathbb{R}^n \text{ such that } \begin{bmatrix} \nabla h_1(x)^\top \\ \vdots \\ \nabla h_m(x)^\top \end{bmatrix} d < 0$$

Necessary condition for Pareto minimizers

Pareto first-order stationary condition: x is a Pareto stationary point if

$$\nexists\, d \in \mathbb{R}^n \text{ such that } \begin{bmatrix} \nabla h_1(x)^\top \\ \vdots \\ \nabla h_m(x)^\top \end{bmatrix} d < 0$$

Or equivalently, if the convex hull of $\nabla h_i(x)$'s contains the origin

$$\exists \lambda \in \Delta^m \text{ such that } \sum_{i=1}^{m} \lambda_i \nabla h_i(x) = 0$$

where $\Delta^m = \{\lambda : \sum_{i=1}^m \lambda_i = 1, \lambda_i \geq 0, \forall i = 1, ..., m\}$ is a simplex set.

Necessary condition for Pareto minimizers

Pareto first-order stationary condition: x is a Pareto stationary point if

$$\nexists\, d \in \mathbb{R}^n \text{ such that } \begin{bmatrix} \nabla h_1(x)^\top \\ \vdots \\ \nabla h_m(x)^\top \end{bmatrix} d < 0$$

Or equivalently, if the convex hull of $\nabla h_i(x)$'s contains the origin

$$\exists \lambda \in \Delta^m \text{ such that } \sum_{i=1}^{m} \lambda_i \nabla h_i(x) = 0$$

where $\Delta^m = \{\lambda : \sum_{i=1}^m \lambda_i = 1, \lambda_i \geq 0, \forall i = 1, ..., m\}$ is a simplex set.

Note: when all the functions are convex, $x \in \mathcal{P}$ iff x is Pareto first-order stationary.

Overview of methods for MOO

Methods with a priori preferences (Ehrgott 2005; Miettinen 2012)

- Weighted-sum method

$$\min\ S(x;a) = \sum_{i=1}^{m} a_i f_i(x), \text{ where } a \in \Delta^m$$

  Limitations: hard to preselect weights for different magnitudes; cannot find Pareto minimizers in non-convex regions.

- ϵ-constraint method

$$\min\ f_i(x) \text{ s.t. } f_j(x) \le \epsilon_j,\ \forall j \neq i$$

  where $\epsilon_j \ge \min_{x\in\mathcal{X}} f_j(x)$ are upper bounds.
  Limitations: inappropriate upper bounds lead to infeasibility.

Common issues: output single nondominated point at one run, produce poorly distributed Pareto front with multiple runs.

Overview of methods for MOO

Methods with a posteriori preferences

Population-based heuristic methods: no convergence proofs, e.g.,

- NSGA-II (genetic algorithms) (Deb et al. 2002).
- AMOSA (simulated annealing) (Bandyopadhyay et al. 2008).

Convergent methods: proved convergence to Pareto stationary points

- multi-gradient method (Fliege and Svaiter 2000).
- Newton's method for MOO (Fliege et al. 2009).
- direct multi-search algorithm (Custódio et al. 2011), etc.

Superiority: able to construct the whole well-spread Pareto front.

"Steepest" common descent direction

The multi-gradient algorithm iterates: $x_{k+1} = x_k + \alpha_k d_k$.

"Steepest" common descent direction

The multi-gradient algorithm iterates: $x_{k+1} = x_k + \alpha_k d_k$.

Subproblem 1 (Fliege and Svaiter 2000):

$$d_k \in \operatorname*{argmin}_{d \in \mathbb{R}^n} \max_{1 \leq i \leq m} \{-\nabla h_i(x_k)^\top d\} + \frac{1}{2}\|d\|^2$$

Note: $d_k = 0 \in \mathbb{R}^n$ iff x_k is Pareto stationary.

"Steepest" common descent direction

The multi-gradient algorithm iterates: $x_{k+1} = x_k + \alpha_k d_k$.

Subproblem 1 (Fliege and Svaiter 2000):

$$d_k \in \operatorname*{argmin}_{d \in \mathbb{R}^n} \max_{1 \le i \le m} \{-\nabla h_i(x_k)^\top d\} + \frac{1}{2}\|d\|^2$$

Note: $d_k = 0 \in \mathbb{R}^n$ iff x_k is Pareto stationary.

Subproblem 2: dual problem of Subproblem 1

$$\lambda_k \in \operatorname*{argmin}_{\lambda \in \mathbb{R}^m} \left\| \sum_{i=1}^{m} \lambda_i \nabla h_i(x_k) \right\|^2 \quad \text{s.t.} \quad \lambda \in \Delta^m$$

Then, $d_k = -\sum_{i=1}^{m} (\lambda_k)_i \nabla h_i(x_k)$ is a common descent direction.

"Steepest" common descent direction

The multi-gradient algorithm iterates: $x_{k+1} = x_k + \alpha_k d_k$.

Subproblem 1 (Fliege and Svaiter 2000):

$$d_k \in \operatorname*{argmin}_{d \in \mathbb{R}^n} \max_{1 \le i \le m} \{-\nabla h_i(x_k)^\top d\} + \frac{1}{2}\|d\|^2$$

Note: $d_k = 0 \in \mathbb{R}^n$ iff x_k is Pareto stationary.

Subproblem 2: dual problem of Subproblem 1

$$\lambda_k \in \operatorname*{argmin}_{\lambda \in \mathbb{R}^m} \left\| \sum_{i=1}^{m} \lambda_i \nabla h_i(x_k) \right\|^2 \quad \text{s.t.} \quad \lambda \in \Delta^m$$

Then, $d_k = -\sum_{i=1}^{m} (\lambda_k)_i \nabla h_i(x_k)$ is a common descent direction.

Note: when $m = 1$, one recovers $d_k = -\nabla h_1(x_k)$.

Presentation outline

Stochastic Multi-Objective Optimization

Consider a stochastic multi-objective problem

$$\begin{array}{ll} \min & F(x) \;=\; (f_1(x), \ldots, f_m(x)) \;=\; (\mathbb{E}[f_1(x;w)], \ldots, \mathbb{E}[f_m(x;w)]) \\ \text{s.t.} & x \in \mathcal{X} \end{array}$$

where $w \in \mathbb{R}^{m \times p}$ are random parameters obeying a certain distribution.

Stochastic Multi-Objective Optimization

Consider a stochastic multi-objective problem

$$\begin{array}{ll} \min & F(x) \;=\; (f_1(x),\ldots,f_m(x)) \;=\; (\mathbb{E}[f_1(x;w)],\ldots,\mathbb{E}[f_m(x;w)]) \\ \text{s.t.} & x \in \mathcal{X} \end{array}$$

where $w \in \mathbb{R}^{m\times p}$ are random parameters obeying a certain distribution.

Subproblem 3: replace $\nabla f_i(x_k)$ by an estimate $g_i(x_k;w_k)$ in Subproblem 2

$$\begin{array}{c} \lambda^g(x_k;w_k) \;\in\; \underset{\lambda\in\mathbb{R}^m}{\operatorname{argmin}} \left\| \sum_{i=1}^m \lambda_i g_i(x_k;w_k) \right\|^2 \\ \text{s.t.} \;\; \lambda \in \Delta^m \end{array}$$

$g(x_k;w_k) = \sum_{i=1}^m (\lambda_k^g)_i g_i(x_k;w_k)$ denotes the stochastic multi-gradient.

Stochastic Multi-Objective Optimization

Consider a stochastic multi-objective problem

$$\begin{array}{ll} \min & F(x) \; = \; (f_1(x), \ldots, f_m(x)) \; = \; (\mathbb{E}[f_1(x; w)], \ldots, \mathbb{E}[f_m(x; w)]) \\ \text{s.t.} & x \in \mathcal{X} \end{array}$$

where $w \in \mathbb{R}^{m \times p}$ are random parameters obeying a certain distribution.

Subproblem 3: replace $\nabla f_i(x_k)$ by an estimate $g_i(x_k; w_k)$ in Subproblem 2

$$\lambda^g(x_k; w_k) \; \in \; \underset{\lambda \in \mathbb{R}^m}{\operatorname{argmin}} \; \left\| \sum_{i=1}^{m} \lambda_i g_i(x_k; w_k) \right\|^2 \quad \text{s.t.} \;\; \lambda \in \Delta^m$$

$g(x_k; w_k) = \sum_{i=1}^{m} (\lambda_k^g)_i g_i(x_k; w_k)$ denotes the stochastic multi-gradient.

The stochastic multi-gradient (SMG) algorithm: $x_{k+1} = x_k - \alpha_k g(x_k; w_k)$

The stochastic multi-gradient method

Consider using an orthogonal projection P when minimize over a closed and convex set $\mathcal{X} \subseteq \mathbb{R}^n$.

The stochastic multi-gradient method

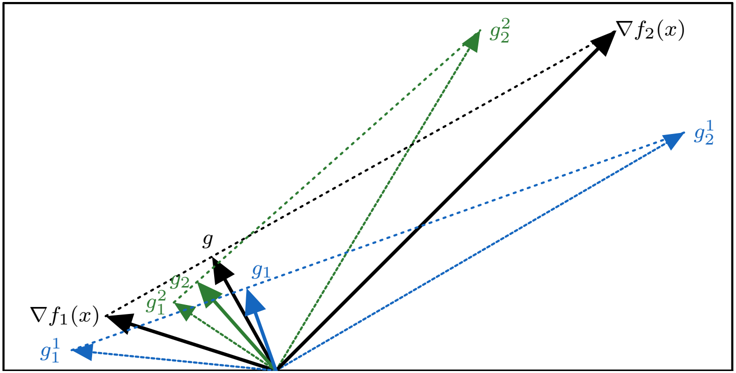
Consider using an orthogonal projection P when minimize over a closed and convex set $\mathcal{X} \subseteq \mathbb{R}^n$.

Algorithm 1 Stochastic Multi-Gradient (SMG) Algorithm

```
1: Choose an initial point $x_0 \in \mathbb{R}^n$ and a step-size sequence $\{\alpha_k\}_{k \in \mathbb{N}} > 0$.
2: for $k = 0, 1, \ldots$ do
3:     Compute the stochastic gradients $g_i(x_k; w_k)$ for $i = 1, \ldots, m$.
4:     Solve Subproblem 3 to obtain the stochastic multi-gradient
           $$g(x_k; w_k) = \sum_{i=1}^{m} (\lambda_k^g)_i g_i(x_k; w_k), \text{ with } \lambda_k^g \in \Delta^m.$$
5:     Update the next iterate $x_{k+1} = P_{\mathcal{X}}(x_k - \alpha_k g(x_k; w_k))$.
6: end for
```

Subproblem 3 illustration

Consider the case $m = 2, n = 2$.



g_1 and g_2 are two stochastic multi-gradients by solving Subproblem 3.

They are estimates of the true multi-gradient g (Subproblem 1 or 2).

Biasedness of the stochastic multi-gradient

Denote

$$\begin{aligned} S(x;\lambda) &= \sum_{i=1}^m \lambda_i f_i(x) \\ \nabla_x S(x;\lambda) &= \sum_{i=1}^m \lambda_i \nabla f_i(x) \end{aligned}$$

Biasedness of the stochastic multi-gradient

Denote

$$\begin{aligned} S(x;\lambda) &= \textstyle\sum_{i=1}^m \lambda_i f_i(x) \\ \nabla_x S(x;\lambda) &= \textstyle\sum_{i=1}^m \lambda_i \nabla f_i(x) \end{aligned}$$

Even under the classical assumption in SG

$$\mathbb{E}_w[g_i(x;w)] = \nabla f_i(x),\ \forall i = 1, \ldots, m$$

it turns out that $g(x;w)$ is a biased estimate, i.e.,

$$\mathbb{E}_w[g(x;w)] \neq \nabla_x S(x;\lambda)$$

where λ are the true coefficients from Subproblem 2.

Biasedness of the stochastic multi-gradient

Denote

$$\begin{aligned} S(x;\lambda) &= \textstyle\sum_{i=1}^m \lambda_i f_i(x) \\ \nabla_x S(x;\lambda) &= \textstyle\sum_{i=1}^m \lambda_i \nabla f_i(x) \end{aligned}$$

Even under the classical assumption in SG

$$\mathbb{E}_w[g_i(x;w)] = \nabla f_i(x), \ \forall i = 1, \ldots, m$$

it turns out that $g(x;w)$ is a biased estimate, i.e.,

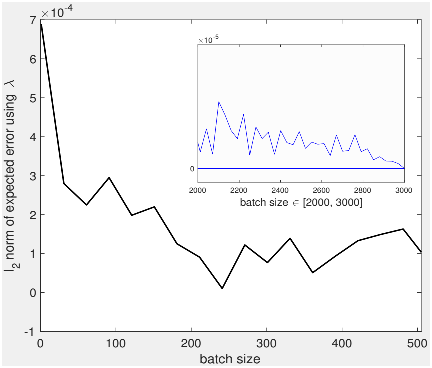
$$\mathbb{E}_w[g(x;w)] \neq \nabla_x S(x;\lambda)$$

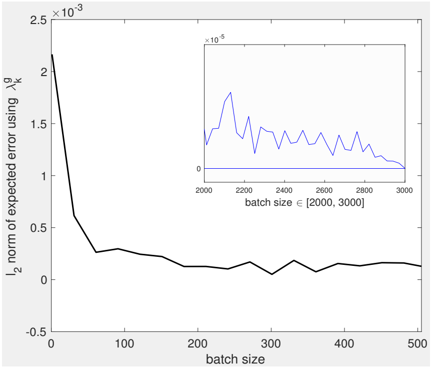
where λ are the true coefficients from Subproblem 2. We also have

$$\mathbb{E}_w[g(x;w)] \neq \mathbb{E}_w[\nabla_x S(x;\lambda^g)]$$

Biasedness illustration

The biasedness using either the true coefficients λ or λ^g decreases as batch size increases and eventually vanishes in the full batch setting.





Assumptions for convergence

1. All objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$ are continuously differentiable with Lipschitz continuous gradients ∇f_i.

2. The feasible region $\mathcal{X} \subseteq \mathbb{R}^n$ is bounded.

3. (Unbiasedness) $\mathbb{E}_w[g_i(x;w)] \; = \; \nabla f_i(x)$.

4. (Bound on the biasedness) There exist $M_1, M_F > 0$ such that

$$\|\mathbb{E}_w[g(x;w) - \nabla_x S(x;\lambda^g)]\| \;\leq\; \alpha\,(M_1 + M_F\|\mathbb{E}_w[\nabla_x S(x;\lambda^g)]\|)$$

   (can be guaranteed by dynamic sampling)

5. (Bound on the second moment) There exist $G, G_V > 0$ such that

$$\mathbb{E}_w[\|g(x;w)\|^2] \;\leq\; G^2 + G_V^2\|\mathbb{E}_w[\nabla_x S(x;\lambda^g)]\|^2$$

Sublinear rate in the strongly convex case

Theorem (Individual f_i are strongly convex (c is max of constants))

Assume $\lambda_t \to \lambda_*$. Let $x_* \in \mathcal{P}$ be associated with λ_*.

Considering a diminishing step-size sequence $\alpha_t = \frac{2}{c(t+1)}$, we have

$$\min_{t=1,\ldots,k} \mathbb{E}[S(x_t;\lambda_t)] - \mathbb{E}[S(x_*;\bar{\lambda}_k)] \;\leq\; \mathcal{O}(1/k)$$

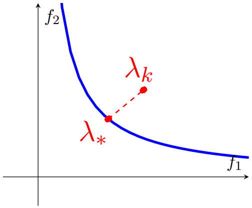
where $\bar{\lambda}_k = \sum_{t=1}^{k} \frac{t}{\sum_{t=1}^{k} t} \lambda_t \in \Delta^m$.

Then, we have $\min_{1\leq t\leq k} \mathbb{E}[S(x_t;\lambda_t)] \to \mathbb{E}[S(x_*;\lambda_*)]$.

Rate in the strongly convex case: stronger assumption

Assume λ_k being a better approximation to λ_*.

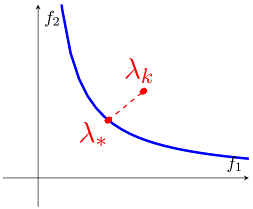
We know $\nabla_x S(x_*; \lambda_*)^\top (x_k - x_*) \geq 0$.



Rate in the strongly convex case: stronger assumption

Assume λ_k being a better approximation to λ_*.

We know $\nabla_x S(x_*; \lambda_*)^\top (x_k - x_*) \geq 0$.



Assumption

For any $x_k \in \mathcal{X}$, one has

$$\nabla_x S(x_*; \lambda_k)^\top (x_k - x_*) \geq 0$$

where λ_k are the true coefficients by solving Subproblem 2.

Rate in the strongly convex case: stronger assumption

Theorem (SL&LNV 2019)

Let x_* be the Pareto minimizer associated with λ_*.

Considering a step-size sequence $\alpha_k = \gamma/k$ with $\gamma > 1/2c$, we have

$$\mathbb{E}[\|x_k - x_*\|^2] \;\leq\; \mathcal{O}(1/k)$$

and

$$\mathbb{E}[S(x_k;\lambda_*)] - \mathbb{E}[S(x_*;\lambda_*)] \;\leq\; \mathcal{O}(1/k)$$

Presentation outline

Pareto-Front stochastic multi-gradient method

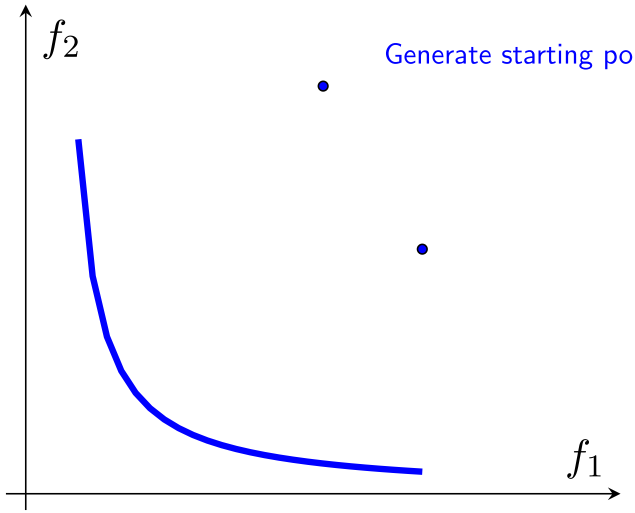
The Pareto-Front version of SMG is designed to obtain a complete Pareto front in a single run.

Algorithm 2 Pareto-Front Stochastic Multi-Gradient (PF-SMG) Algorithm

```
 1: Generate a list of starting points $\mathcal{L}_0$. Select $r, p, q \in \mathbb{N}$.
 2: for $k = 0, 1, \ldots$ do
 3:     Set $\mathcal{L}_{k+1} = \mathcal{L}_k$.
 4:     for each point $x$ in the list $\mathcal{L}_{k+1}$ do
 5:         for $t = 1, \ldots, r$ do
 6:             Add $x + w^t$ to the list $\mathcal{L}_{k+1}$ where $w^t$ is a realization of $w_k$.
 7:     for each point $x$ in the list $\mathcal{L}_{k+1}$ do
 8:         for $t = 1, \ldots, p$ do
 9:             Apply $q$ iterations of the SMG algorithm starting from $x$.
10:             Add the final output point $x_q$ to the list $\mathcal{L}_{k+1}$.
11:     Remove all the dominated points from $\mathcal{L}_{k+1}$.
```

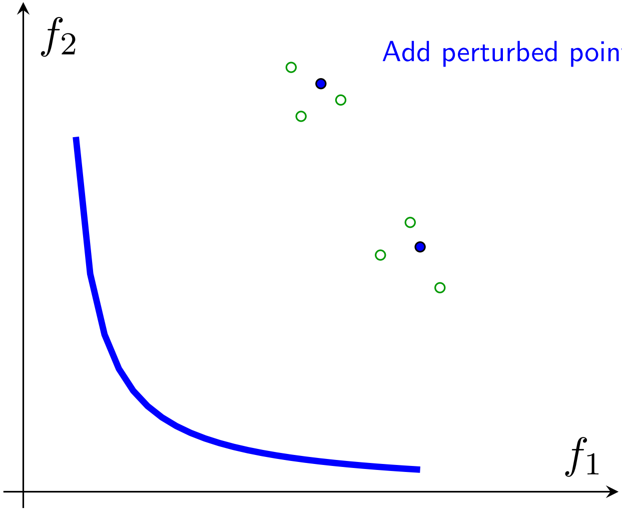
PF-SMG illustration

$r = 3, p = 2$



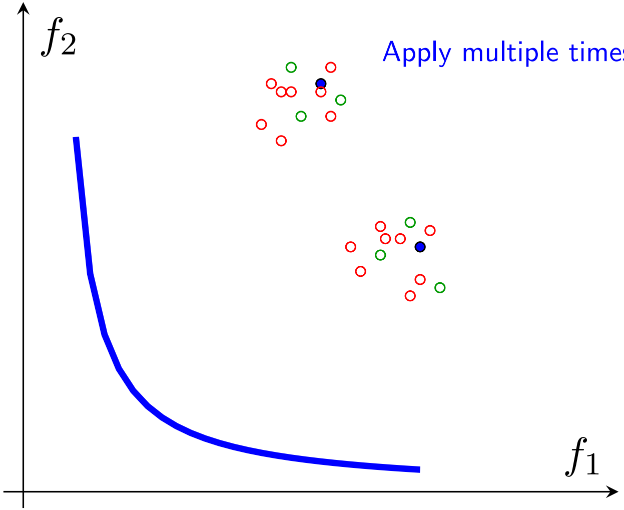
PF-SMG illustration

$r = 3, p = 2$



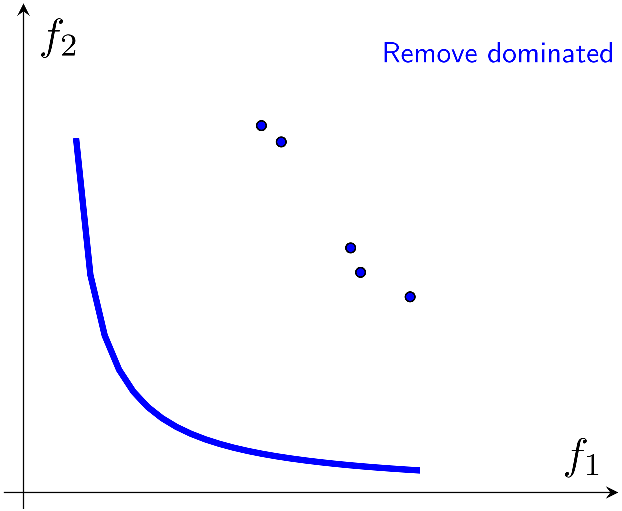
PF-SMG illustration

$r = 3, p = 2$



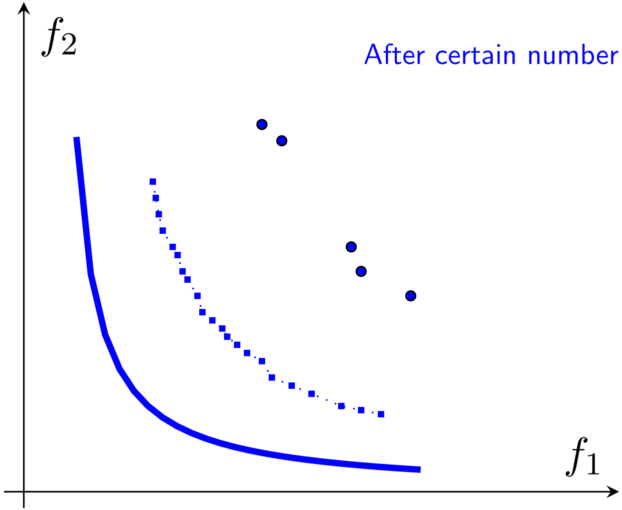
PF-SMG illustration

$r = 3, p = 2$



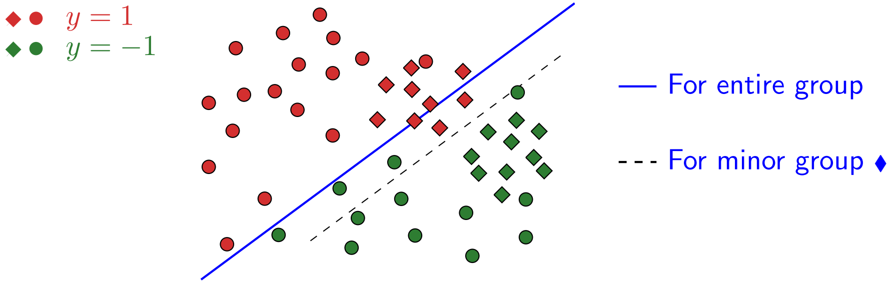
PF-SMG illustration

$r = 3, p = 2$



Numerical results: logistic regression problems

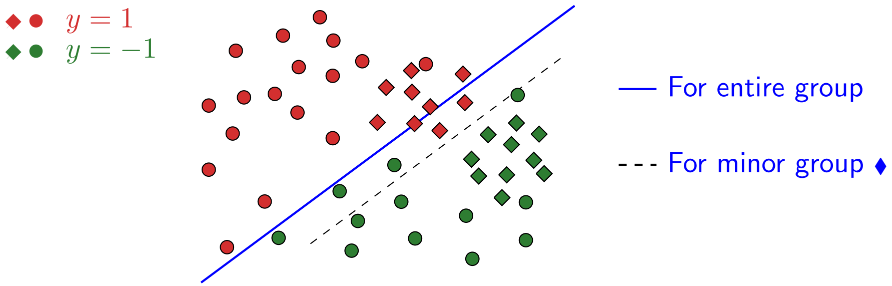
Motivation: consider a set of data points in 2D



Data points labeled by ♦ and ● may be collected from different sources/groups.

Numerical results: logistic regression problems

Motivation: consider a set of data points in 2D



Data points labeled by ♦ and • may be collected from different sources/groups.

Key idea: design a MOO problem to identify the existence of bias in data and define the best trade-off if data bias exists.

- Recall the logistic (prediction) loss function

$$f(r,b) \;=\; \frac{1}{N}\sum_{j=1}^{N}\log(1+e^{-y_j(r^\top a_j-b)})+\frac{\lambda}{2}\|r\|^2$$

where $\{(a_j,y_j)\}_{j=1}^N$ are i.i.d. feature/label pairs sampled from a certain joint probability distribution of (A,Y).

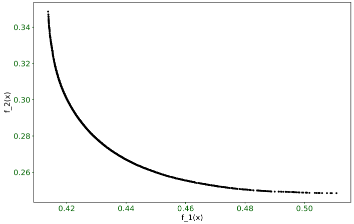
- Testing data sets are selected from LIBSVM (Chang and Lin 2011).
- Split a data set into two groups according to a binary feature. Let J_1 and J_2 be two index sets. A two-objective problem is constructed as

$$\min_{r,b}\,(f_1(r,b),f_2(r,b))$$

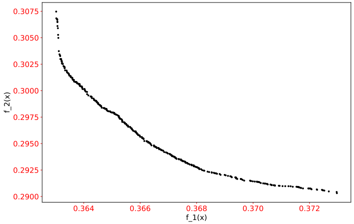
where

$$f_i(r,b) \;=\; \frac{1}{|J_i|}\sum_{j\in J_i}\log(1+e^{-y_j(r^\top a_j-b)})+\frac{\lambda_i}{2}\|r\|^2$$

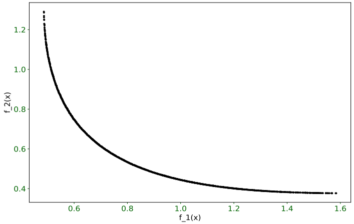
Approximated Pareto fronts for the multi-objective logistic regression problems:



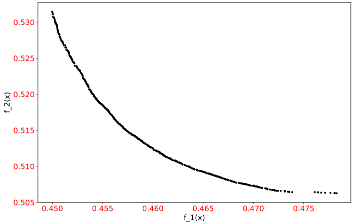
(a) *heart*



(b) *australian*



(c) *svmguide3*



(d) *german.numer*

Consistently, wider Pareto fronts of *heart* and *svmguide3* indicate higher distinction between two groups.

Consistently, wider Pareto fronts of *heart* and *svmguide3* indicate higher distinction between two groups.

Two implications:

- Given groups of data instances for the same problem, one can evaluate the bias by observing the range of Pareto fronts.
- New data instance (of unknown group) can be classified more accurately by selecting a set of nondominated points.

Presentation outline

Conclusions

- Established sublinear convergence rates, $\mathcal{O}(1/k)$ for strongly convex and $\mathcal{O}(1/\sqrt{k})$ for convex case in terms of a weighted sum function.
- Designed PF-SMG algorithm that is robust and efficient to generate well-spread and sufficiently accurate Pareto fronts.
- In logistic binary classification, developed a novel tool for identifying bias among potentially different sources of data.

S. Liu and L. N. Vicente, *The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning*, ISE Technical Report 19T-011, Lehigh University.

Future directions

- Have a (probabilistic?) result for determining the whole Pareto front.
- Investigate variance reduction techniques.
- Deal with nonconvexity, nonsmoothness, general constraints, . . .
- Expand use of MOO in machine learning:
  - Handling discrimination and unfairness (Calders et al. 2009; Hardt et al. 2016).
  - Conflicting robotic learning.