Inverse Parametric Optimization with Partial Examples

Motivation

The motivation for this problem comes from Pairwise Protein Sequence Alignment in BioInformatics: You have two strings of proteins sequences A and B, and you want to align them so that identical or similar characters are aligned in successive columns. (Draw example on board) The way it is calculated is by assigning a score to each tranformation (insertion, deletion, or substitution). There are 20 amino acids, so a 210 substitution scores. You also have gap-open and gap-extension penalties. The total score is calculated by adding up the scores for each transformation. We define an optimal alignment for two sequences to be one where the total score is minimal. Example ACCCGA ACTA ACCCGA AC–TA
Inverse Parametric Optimization is motivated by the following problem. Given a set of input protein sequences and output alignments (that we know to be correct), how do we get the alignment scoring scheme (or the values of the substitution scores and the gap-open and gap-extension penalties) that make each alignment be the optimal output for each input. This is problem is called Inverse Alignment. Inverse Parametric Optimization a way of solving the general problem.

Inverse Parametric Optimization with Complete Examples

Let f be a scoring function with parameters $p = (p_1, p_2, ..., p_t)$. Let

$$f_p(O) = \sum_{1<=i<=t} p_i g_i(O) + g_0(O)$$

.
Forward Optimization Problem: Given input I, p, find legal output O that minimizes $f_p(O)$.
Inverse Optimization problem: Given input pair I, O, find parameter values $p = (p_1, p_2, ...p_t)$ such that $f_p(O)$ is minimum over all legal outputs for I.

We will call the pair (I, O) an example. What does it mean for O to be optimal? it means for all other legal ouputs O' for I, $f_p(O) <= f_(O')$. For our problem, we can generalize to a set of examples:
$\{(I_1, O_1, (I_2, O_2)...(I_k, O_k)\}$. Then we need to find parameter choice $p*$ such that O_1 is optimal for input I_1 under f_p*, O_2 is optimal for input I_2 under f_p* etc. Generalize to finding one parameter choice $p*$ that makes each O_i be as close to optimal as possible for its input I_i.
Find $x*$ that minimizes average absolute error: Let $x* = argminE(x)$, $x \in D$ where

$$E(x) = \frac{1}{k} \sum_{1<=i<=k} (f_x(O_i) - f_x^*(I_i)$$

where $f_x^*(I_i))$ is the value of optimal solution under f_x for input I_i.

We can reduce the problem to a Linear Programming Problem. Linear programming optimizes a linear function of real variables over a domain given by linear inequalities. Geometrically this domain, which is an intersection of half-spaces, is a convex body called a polyhedron. A linear programming problem is usually defined with a) Objective function f (that we want to maximize, or minimize), and b) the constraints that the solution needs to satisfy.

Reduction to Linear Programming problem

Let t be the number of parameters, let k be the number of examples. Real-valued variables: $x = (x_1, x_2, ...x_t)$ (one for each parameter) and $\delta = (\delta_1, \delta_2, ...\delta_k)$, one for each example.
Domain inequalities, Infinite number of inequalities: For all examples: For all legal outputs P for input I_i $f_x(O_i) - f_x(P) <= \delta_i$
Objective function for LP program:
min $\frac{1}{k} \sum_{1<=i<=k} \delta_i$.
Claim that optimal solution to linear programming problem $x*, P*$ gives an optimal parametric choice $x*$ for our inverse parametric optimization problem:
1) $\delta_i >= max f_x(O_i) - f_x(P)$ over legal P. 2) minimizing $\frac{1}{k} \sum \delta_i$ makes $\delta_i = max$.
If δ_i is max, then we minimize the error.

Solving LP by separation

One of the truly far-reaching results in linear programming is what we call the Separation Theorem. The optimization problem for a rational polyhedron $P \subset Rd$ is: Given rational coefficients c that specify the objective function, find a point $x \in P$ that minimizes the objective function, or determine that P is empty. The separation problem for P is: Given a point $y \in Rd$, either (1) find rational coefficients w and b that specify an inequality such that $wx \leq b$ for all $x \in P$, but $wy > b$; or (2) determine that $y \in P$. In other words, a separation algorithm that solves the separation problem for polyhedron P determines whether point y lies inside P, and if it lies outside, finds an inequality that is satisfied by all points in P but is violated by y. Such a violated inequality gives a hyperplane that separates y from P. The Separation Theorem says that, Theorem 1 (Equivalence of Separation and Optimization [6, 14, 10]). The optimization problem on a bounded rational polyhedron can be solved in polynomial time if and only if the separation problem can be solved in polynomial time.
For an example (I_i, O_i), and a parameter choice, we can find out if an inequality is violated in the following manner. Compute an optimal alignment given the input (this is solving the forward optimization problem) and the parameter choice $P* = F(I_i, x)$. 2) Test whether $f_x(O_i) - f_x(p*) <= \delta_i$. The left-hand side is going to be maximum over all other alignments.
Cutting Plane algorithm:
1) Start with a small subset P of the inequalities in the linear program (domain inequalities). 2. Compute an optimal solution $\bar{x}$ to the linear program given by subset P. If no such solution exists, halt and report that L is infeasible. 3. Call the separation algorithm for L on $\bar{x}$. If the algorithm reports that $\bar{x}$ satisfies L, output $\bar{x}$ and halt: $\bar{x}$ is an optimal solution for L. 4. Otherwise, add to P the violated inequality returned by the separation algorithm in step 3, and loop back to step 2.
Thus at the end of the cutting plane algorithm we have a solution to the Inverse Optimization Problem.
(Maybe?) User provides input type, output type, domain inequalities, vector of examples, forward solver, and param coefficients.

Optimizations to the algorithm

1) Have user provide an initial set of solutions, and add the violated inequalities to the initial set of inequalities in the LP. 2) Maintain an ordered queue Q of examples, whenever an example is satisfied, it is moved to end of the Q (so the next time it is not the first to be tested). 3) Once the number of inequalities in current LP exceeds bt, we shrink the LP by keeping only the at tightest inequalities (culling).

Extending to Partial Examples

Inverse Parametric Optimization with Partial examples is also motivated by alignment. Often times we have output alignments with some regions that are unspecified or unreliable. For the general case, we defined partial examples to simply be examples where only some part of the output is reliable.

Definition: Given a partial example A, and a parameter choice $x*$, a completion $\bar{A}$ is one that agrees on the reliable columns of A. For alignments, a completion for a partial example is found by calling the Forward solver on the non-reliable parts, and stitching them together with the reliable parts (draw diagram).

Discrepancy criterion

For the partial examples, we will use a new objective function. Instead of minimizing the average absolute error, we want to minimize discrepancy, which is defined as:

$$D(x) = \frac{1}{k} \sum_{1 \leq i \leq k} \max_{B_i of S_i} \left\{ (1-\alpha) \frac{f_x(A_i) - f_x(B_i)}{L(S_i)} + \alpha d(A_i, B_i) \right\}$$

The maximum is over all outputs B_i for a given input S_i. So our output is the parameter choice $x*$ that minimizes discrepancy over all examples. When $\alpha = 0$, we have the average absolute error approach. $L(S_i)$ is the input, and $d(A_i, B_i)$ is the recovery error, or in other words, a measure for the number of reliable portion from A that was recovered in B. d ensures that the parameter choice is such that it forces the reliable columns to be the same.
In our interface user provides: input type (should have length), partial output type which tells us which part is reliable and also a distance function, a Discrepancy solver (which returns the output that maximizes discrepancy for a given input and parameter choice), a completion, and ParamCoeffs. User can specify the number of iterations it should go on for. User can also provide an initial parameter choice $\tilde{x}$.

Iterative Algorithm

Start with an initial completion $(A_i)_0$ for each partial example A_i, which may be formed by calling the user-provided completion method with the default parameter choice $(x)_0$. (Or, can use the trivial completion). Then iterate the following process for j = 0, 1, 2, ... Compute an optimal parameter choice $(x)_{j+1}$ by solving inverse optimization on the complete examples $(A_i)_j$ (this solver is slightly different, see below). Given (x)j+1, form a new completion $(A_i)_{j+1}$ of each example A_i by calling the user-provided Completion function. Keep doing this until $j =$ the number of iterations the user specified, or the error is small enough.
Variation to the Inverse Parametric Solver: 1) The SepAlg calls DiscrepancySolver instead of the ForwardSolver. Our new inequality is:

$$D(x) = (1-\alpha) \frac{f_x(A_i) - f_x(B_i)}{L(S_i)} + \alpha d(A_i, B_i) \leq \delta_i$$

Again, the left hand side is the maximum it can be, since our discrepancy solver return B_i with maximum discrepancy. Our new objective function is:

$$\frac{1}{k} \sum_{1 \leq i \leq k} \delta_i$$

2) Uses the solution in each outer iteration as initial parameter choice to provide speed up to the Inverse Parametric Solver.
Theorem from "Learning Scoring Schemes for Sequence Alignment from Partial Examples":
For the iterative scheme for inverse alignment from partial examples, denote the error in score for iteration $j \geq 1$ by

$$e_j := D(x^{(j)}, \bar{A}(j-1)$$

where the right-hand side measures discrepancy for the given parameters on the given completions. Then,

$$e_1 \geq e2 \geq e3 \geq ... \geq e*$$

where e* is the optimum discrepnacy for inverse optimization from partial examples A_i under the discrepancy criterion.