Diabetes Prediction with Incomplete Patient Data

Hao Yi Ong, Dennis Wang, Xiao Song Mu

CS 221 Aritificial Intelligence: Principles and Techniques Class Project

Introduction

- Given a set of electronic health records, we want to have a smart predictor that prompts high-risk patients to obtain Type II Diabetes testing

- Original Kaggle challenge:
  - Patients all have a standard database and a full medical record
  - I.e., exact same tests taken, same variables recorded, etc.
- In practice, however:
  - Not everyone has taken the same tests and gotten regular checkups
  - Database inconsistencies or errors in inputting data may exist
- Predictor must be able to accurately classify based on incomplete or erroneous medical records to be useful

Diabetes Prediction

- Given
  - *Training set* containing standardized patient medical records
  - *Testing set* containing patient medical records with missing information and unknown erroneously recorded data
- Output
  - *Bayesian network structure* encoding the conditional dependencies between medical record variables
  - *Bayesian network parameters* encoding the conditional probabilities for each variable
  - *Probabilistic inference* on the learned Bayesian network for classification
- To minimize the error rate, including false positives and false negatives, on classifying whether a patient has Type II Diabetes

Evaluation Criteria

- *Baseline* of logistic regression
  - Feature vector with basic data including height, weight, body mass index...
  - Cross validation with 10%-hold-out on the training data
  - Obtained a false positive rate of 0.7% and false negative rate of 15.3% for a total error rate of 16% (84% accuracy)
- *Oracle*
  - Ideal oracle would be experienced physicians who have correctly advised patients to test for diabetes given their medical history
  - Infeasible time-wise and financially for the scope of our project
  - Use as surrogate measures the test accuracies of established diabetes tests vetted by the US Department of Health and Human Services
  - Specifically, HBA1c, FPG, and OGTT, which have 85–95% accuracy

Problem Formulation

The design variables for our truss optimization are:

- *Cross sectional areas* $a \in \mathbf{R}^m$, where $a_i \in \mathbf{R}$ is the area of the i^{th} bar
- *Coordinates* $x \in \mathbf{R}^{2n}$, where $x_j \in \mathbf{R}^2$ are the coordinates of the j^{th} node

Our problem data are:

- *Loading forces* $F \in \mathbf{R}^{2n}$, where $F_j \in \mathbf{R}^2$ is the load on the j^{th} node
- *Material densities* $\rho_1, \ldots, \rho_m \in \mathbf{R}$ of bars
- *Young's moduli* $E_1, \ldots, E_m \in \mathbf{R}$ characterizing the elasticities of the bars
- *Bar lengths* $L_1, \ldots, L_m \in \mathbf{R}$, which are dependent on node coordinates x
- *Force mapping matrix* $P(\mathcal{X}) \in \mathbf{R}^{m \times 2n}$, which relates loads F to the internal stresses experienced by the bars, $f \in \mathbf{R}^m$; implicit in P is an adjacency matrix relating each bar to its attachment points
- *Stiffness matrix* $K(\mathcal{X}, a, L)$, which determines the amount of flex in the truss

$$K = \sum_{i=1}^{m} \frac{E_i a_i}{L_i^2} p_i p_i^T,$$

where $p_1, \ldots, p_m$ are the columns of the force mapping matrix P

Our truss design optimization is further characterized by the following variables, whose relations contain all of the physics of the problem:

- *Node deflections* $u \in \mathbf{R}^{2n}$ due to the truss flexing under loads F, where $u_j \in \mathbf{R}^2$ is the deflection of the j^{th} node; by Hooke's Law, we have the force balance $F = Ku$
- *Internal stress* $f_i \in \mathbf{R}$ experienced by each bar due to the node deflections

$$f_i = -\frac{E_i a_i}{L_i^2} p_i^T u, \quad i = 1, \ldots, m$$

- *Stored elastic energy* $\Theta = \frac{1}{2} F^T u$, which we minimize in order to maximize the truss stiffness

An Alternating Convex Optimization Approach

The minimization of Θ in (a, x) that follows from our formulation above is non-convex. As a heuristic to solve the optimization problem, we first optimize over the bar sizes a, and then over the node coordinates x:

- We perform a linear change of coordinates to cast the bar sizing problem as a second-order cone program (SOCP) in $w, v \in \mathbf{R}^m$:

$$w_i + v_i = -\frac{1}{2} \left(u^T P\right)_i f_i,$$

$$w_i - v_i = a_i$$

  The value of $w_i + v_i$ is therefore the spring energy stored in the i^{th} bar
- Holding x constant, find the bar cross sectional areas a that minimize Θ:

$$\begin{array}{ll} \text{minimize} & \Theta = 1^T (w + v) \\ \text{subject to} & Pf + F = 0 \\ & M (w - v) \leq d \\ & \left\| \left(v_i, \frac{L_i}{\sqrt{E_i a_i}} f_i \right) \right\|_2 \leq w_i, \quad i = 1, \ldots, m \end{array} \qquad (1)$$

(Cont'd)

- Holding a constant, find a set of displacements $y \in \mathbf{R}^{2n}$ that "shift" the node coordinates x from their original positions and minimize Θ:

$$\begin{array}{ll} \text{minimize} & \Theta = 1^T (w + v) \\ \text{subject to} & Pf + F = 0 \\ & \frac{1}{2} ((w_i - v_i) - 1) = \frac{p_i^T y_i}{L_i}, \quad i = 1, \ldots, m \\ & \left\| \left(v_i, \frac{L_i}{\sqrt{E_i a_i}} f_i \right) \right\|_2 \leq w_i, \quad i = 1, \ldots, m \\ & \|y_i\|_2 \leq \epsilon_i, \quad i = 1, \ldots, m \\ & g(y) = 0, \end{array} \qquad (2)$$

  where $g(y) = 0$ enforces truss symmetry, and $\|y_i\|_2 \leq \epsilon_i$ restrict node shifts.

In our heuristic, we first discretize the physical space as in traditional approaches to obtain $\hat{\mathcal{D}}$, and alternate between solving (1) and (2) in each iteration:

> **given** $\mathcal{X}^{\text{fixed}}$, $\mathcal{F}$, $\hat{\mathcal{D}}$
>
> Generate set of node coordinates x^0 from $\hat{\mathcal{D}}$, set $x := x^0$
>
> **repeat**
>
> 1. Given x, obtain a and Θ_1 as the solution to and objective of (1)
> 2. Given a, obtain y and Θ_2 as the solution to and objective of (2), set $x := x + y$
> 3. **break if** Θ_1 and Θ_2 converge
>
> **return** a, x

Example: Bridge Design

This problem has 791 variables and 219 constraints. Out of several solvers, SCS was the fastest at 0.3 s per iteration (vs. 2.0 s with SeDuMi at comparable accuracy requirements). SCS's speed advantage scales with problem size (~100 times faster than SeDuMi with 16000 variables, 4000 constraints).

Conclusion

Our alternating convex optimization approach presents a promising tool to solving the non-convex truss design problem. Future work should extend the model to 3-D and compare this approach to other existing methods.

Acknowledgments

We thank Professor Liang and the instructor team, as well as fellow classmates for their help on our project.