Optimization goes Data Science goes Optimization

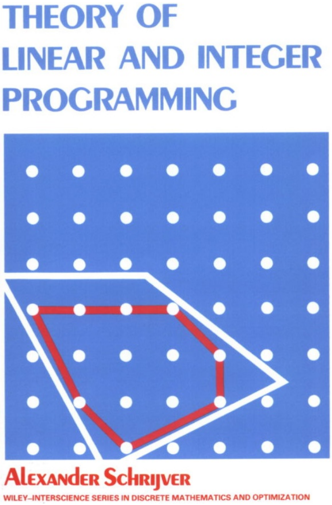
@mluebbecke

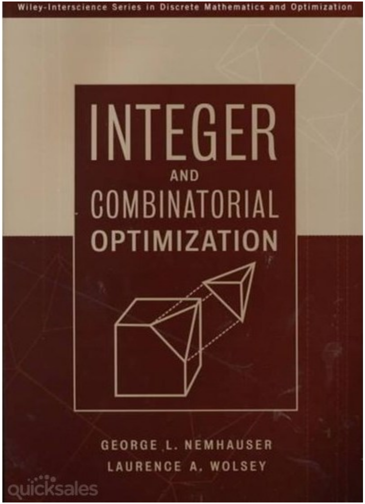
Data Science: Theory & Applications · Oct 26, 2015

Research Area Operations Research and Management | RWTH AACHEN UNIVERSITY

Operations Research | RWTH AACHEN UNIVERSITY

If you need to reduce us to one label





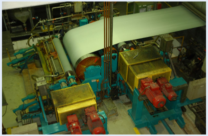
Example applications

a lot!


vehicle routing


container logistics


course timetabling


production planning


materials stacking


patient scheduling

Our main machinery

schedule course c at period t in room r, or not

$$
\begin{array}{rrcll}
 & \min \sum_{c,t,r} prio(c,t) \cdot x_{c,t,r} & & & \\
s.t. & \sum_{t \in T(c), r \in R(c)} x_{c,t,r} & = & n(c) & \text{courses } c \\
 & \sum_{c \in R^{-1}(r)} x_{c,t,r} & \leq & 1 & \text{periods } t, \text{ rooms } r \\
 \sum_{r \in R(c_1)} x_{c_1,t_1,r} + & \sum_{r \in R(c_2)} x_{c_2,t_2,r} & \leq & 1 & \text{conflicts} \\
 & x_{c,t,r} & \in & \{0,1\} &
\end{array}
$$

a naïve integer program, a.k.a. “three-indexed”

Integer programming: Progress on algorithms

- very effective algorithm: branch-and-cut
- industry strength implementations available ("solvers")
- development since 1991
  - computer speedup: factor 2 000
  - algorithmic speedup: factor 500 000

$\Rightarrow$ easily solve problems with 10^6 variables and 10^5 constraints

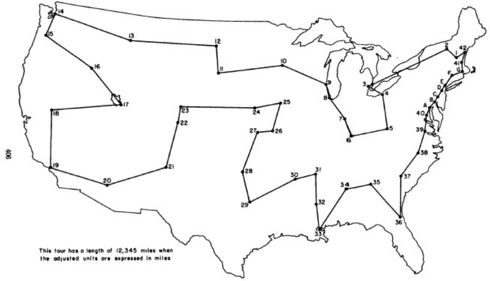
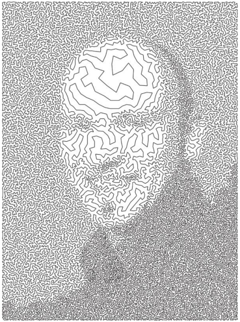
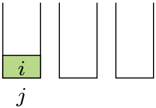


FIG. 16 The optimal tour of 49 cities



The bin packing problem

$$
\begin{aligned}
\min \sum_{\text{bins } j} y_j & & \\
\sum_{\text{bins } j} x_{ij} &= 1 && \text{items } i \\
\sum_{\text{items } i} a_i x_{ij} &\leq b && \text{bins } j \\
x_{ij} &\leq y_j && i, j \\
x_{ij},\, y_j &\in \{0, 1\} && i, j
\end{aligned}
$$

$x_{ij} \sim$ 

Performance of the first model

```
   ...                                 ...
   4485  2808      45.2933   138       47.0000       45.2933   256083    3.63%
   4558  2865      46.0000    53       47.0000       45.2933   260817    3.63%
   4618  2911      46.0000    56       47.0000       45.2933   265554    3.63%
   4679  2946      45.3133   100       47.0000       45.2933   270877    3.63%
   4759  3026      45.3133    54       47.0000       45.2933   274429    3.63%
   4859  3126      45.3667    46       47.0000       45.2933   278978    3.63%
   4957  3224      45.3667    55       47.0000       45.2933   284217    3.63%
   5067  3304      45.9800    30       47.0000       45.2933   289414    3.63%
   5160  3383      45.3133    68       47.0000       45.2933   292707    3.63%
   5317  3526      45.2933   147       47.0000       45.2933   297019    3.63%
Elapsed real time =  74.99 sec. (tree size = 61.35 MB, solutions = 7)
   5418  3625      46.0000    67       47.0000       45.2933   301117    3.63%
   5476  3662      45.3133   115       47.0000       45.2933   306847    3.63%
   5550  3726      45.2933   153       47.0000       45.2933   312240    3.63%
   5639  3809      46.0000    42       47.0000       45.2933   316751    3.63%
   5745  3900      46.0000    51       47.0000       45.2933   319540    3.63%
   5801  3948      46.0000    39       47.0000       45.2933   324910    3.63%
   5884  4013      45.3133    62       47.0000       45.2933   329653    3.63%
   ...                                 ...
```

✔ model symmetry is a problem here

Bin packing: A model with "more meaningful" variables

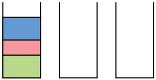
- $P_j \sim$ *all* possible patterns to fill bin j

$$\min \sum_{\text{bins } j} \sum_{p \in P_j} \lambda_{pj}$$

$$\sum_{\text{bins } j} \sum_{p \in P_j : i \in p} \lambda_{pj} = 1 \qquad \text{items } i$$

$$\sum_{p \in P_j} \lambda_{pj} \leq 1 \qquad \text{bins } j$$

$$\lambda_{pj} \in \{0, 1\} \qquad j = 1, \ldots, n,\ p \in P_j$$

$\lambda_{pj} \sim$ 

j

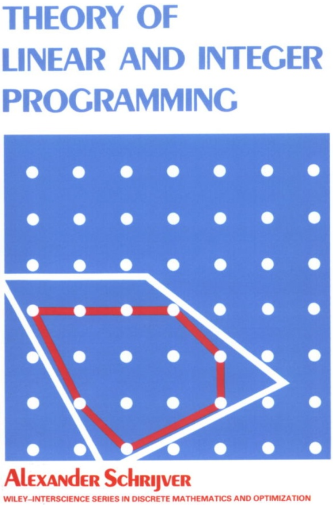
Performance of the second model

```
 time | node  | left  |LP iter|frac |vars |cons |cols |strbr|  dualbound   | primalbound  |  gap
t 0.0s|     1 |     0 |   125 |   0 | 121 | 120 | 121 |   0 |      --      | 1.200000e+02 |    Inf
r 0.0s|     1 |     0 |   125 |   - | 121 | 120 | 121 |   0 |      --      | 1.150000e+02 |    Inf
r 0.0s|     1 |     0 |   131 |   - | 122 | 120 | 122 |   0 |      --      | 1.110000e+02 |    Inf

...

  0.4s|     1 |     0 |  5977 | 100 | 410 | 120 | 273 |   0 |      --      | 9.700000e+01 |    Inf
  0.4s|     1 |     0 |  6793 | 104 | 460 | 120 | 310 |   0 | 4.529333e+01 | 9.700000e+01 | 114.16%
b 0.4s|     1 |     0 |  6793 | 104 | 460 | 120 | 310 |   0 | 4.529333e+01 | 5.800000e+01 |  28.05%
E 0.5s|     1 |     0 |  7575 | 104 | 460 | 120 | 310 |   0 | 4.529333e+01 | 4.800000e+01 |   5.98%
  0.5s|     1 |     0 |  7575 | 104 | 460 | 120 | 310 |   0 | 4.529333e+01 | 4.800000e+01 |   5.98%
  0.7s|     1 |     2 |  7575 | 104 | 460 | 120 | 310 |  42 | 4.529333e+01 | 4.800000e+01 |   5.98%
R 3.6s|    24 |    23 | 16461 |   5 | 895 | 120 | 767 | 447 | 4.529333e+01 | 4.700000e+01 |   3.77%
b 3.6s|    25 |     0 | 16461 |   - | 895 | 120 | 767 | 452 | 4.550000e+01 | 4.600000e+01 |   1.10%

SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 3.64
Solving Nodes      : 25
Primal Bound       : +4.60000000000000e+01 (383 solutions)
Dual Bound         : +4.60000000000000e+01
Gap                : 0.00 %
```

If nothing else, take away this message

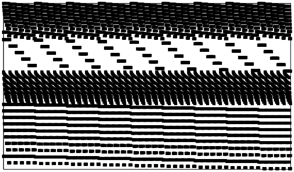
✎ in integer programming, a "good" model is crucial

Revisiting. . .


THEORY OF
LINEAR AND INTEGER
PROGRAMMING
ALEXANDER SCHRIJVER
WILEY–INTERSCIENCE SERIES IN DISCRETE MATHEMATICS AND OPTIMIZATION

Detecting structure in matrices

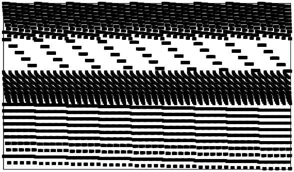
- key to model strengthening “by decomposition”



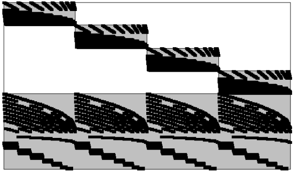
10teams

Detecting structure in matrices

- key to model strengthening "by decomposition"



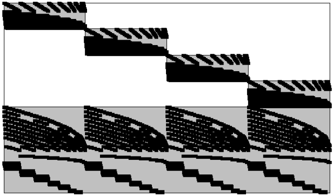
10teams

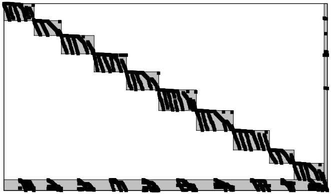


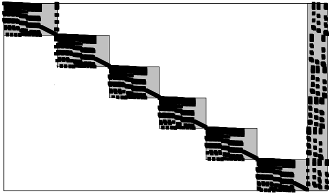
10teams

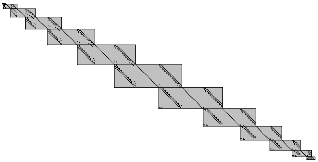
✔ this is graph partitioning/clustering

Detecting structure in matrices

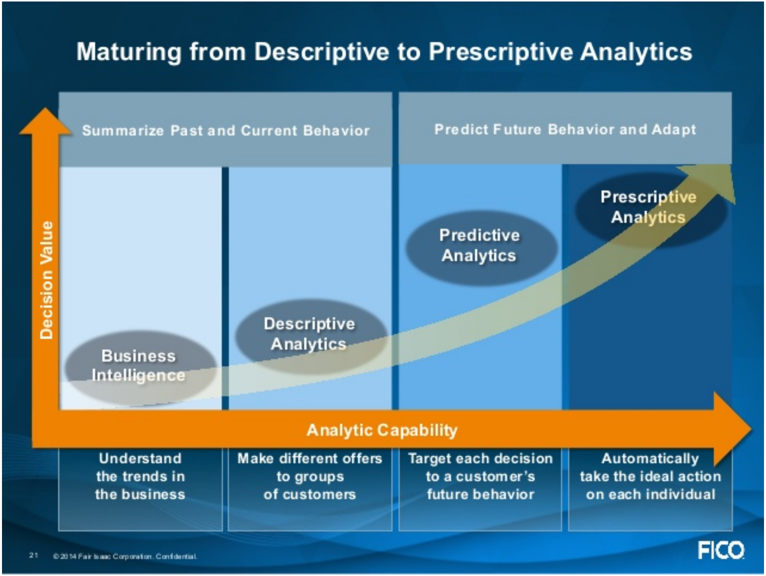




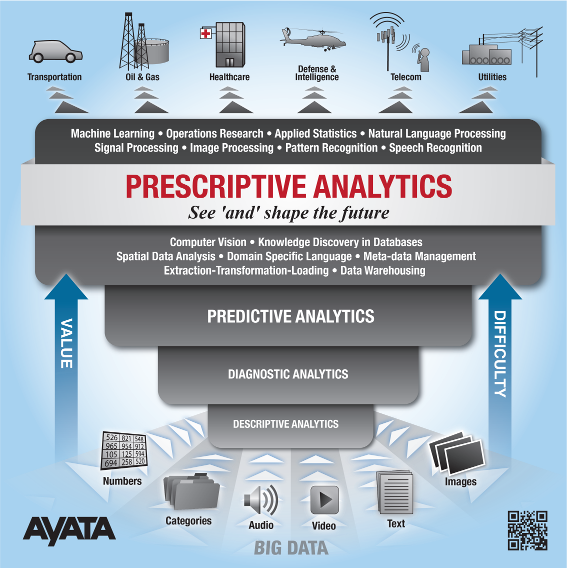




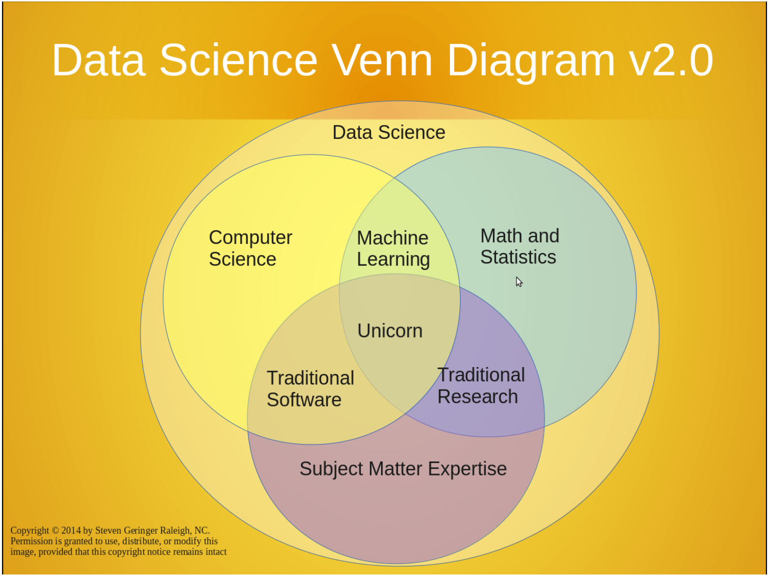
Business perspectives



Business perspectives

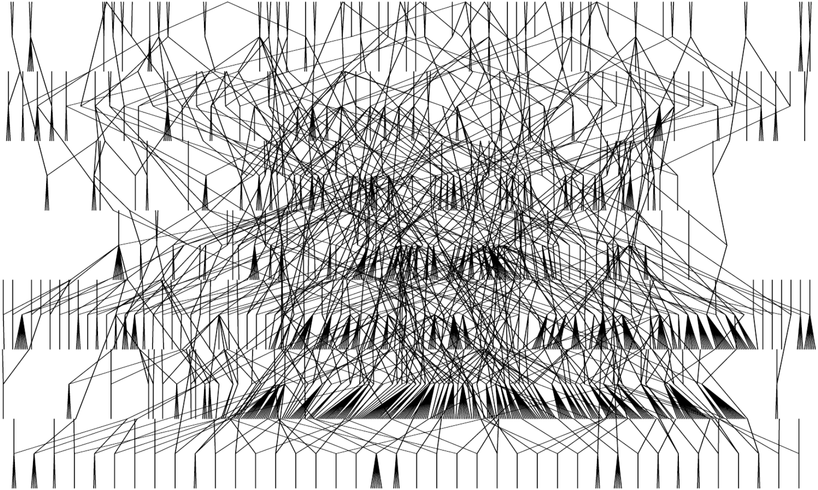


Twitter is full of...



Twitter is full of...

1614 node ‘Polar Eskimo Genealogy’



Training of SVMs

Fast training of Support Vector Machines with Gaussian kernel

Matteo Fischetti
DEI, University of Padova, via Gradenigo 6/A, 35100 Padova (Italy)
e-mail: *matteo.fischetti@unipd.it*

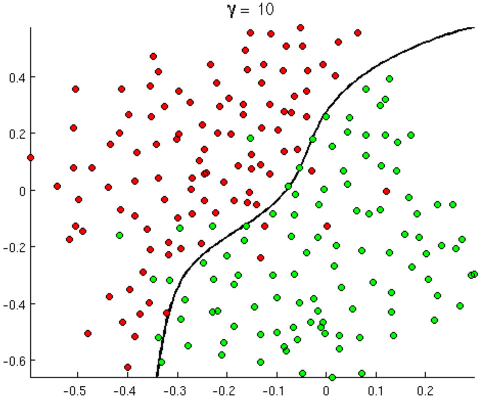
10 July 2014. Revised 12 February 2015

Abstract

Support Vector Machines (SVM's) are ubiquitous and attracted a huge interest in the last years. Their training involves the definition of a suitable optimization model with two main features: (1) its optimal solution estimates the a-posteriori optimal SVM parameters in a reliable way, and (2) it can be solved efficiently. Hinge-loss models, among others, have been used with remarkable success together with cross validation—the latter being instrumental to the success of the overall training, though it can become very time consuming. In this paper we propose a different model for SVM training, that seems particularly suited when the Gaussian kernel is adopted (as it is often the case). Our approach is to model the overall training problem as a whole, thus avoiding the need of cross validation. Though our basic model is an NP-hard Mixed-Integer Linear Program, some variants can be solved very efficiently by simple sorting algorithms. Computational results on test cases from the literature are presented, showing that our training method can lead to a classification accuracy comparable (or even slightly better) than the classical hinge-loss model, with a speedup of 2-3 orders of magnitude.

Keywords: support vector machine, classification, mixed-integer programming.

Evaluation of classification algorithms



Perspectives

- What do we really know about our integer programs?
- What do we really know about our data?
- ...
- ...
- Optimization $\rightarrow$ Data Science
- Optimization $\leftarrow$ Data Science
- Optimization $\leftrightarrow$ Data Science
- ...