Scaling up Generic Optimization

Joachim Giesen Lars Kühne Sören Laue Jens Müller

Friedrich-Schiller-University Jena, Germany

SPP 1736 – ”Algorithms for Big Data”

seit 1558

Scaling up Generic Optimization

- Optimization is ubiquitous in science, engineering, and economics.
- Optimization problems come in many different flavors.

Scaling up Generic Optimization

- Optimization is ubiquitous in science, engineering, and economics.
- Optimization problems come in many different flavors.

- Linear,
- Convex,
- Nonlinear,
- Discrete optimization problems

Scaling up Generic Optimization

We focus on large-scale convex optimization problems.

- Cover many important problems.
- Many discrete optimization problems can be relaxed to convex problems, e.g., max-cut, min-cut
- Use examples from big data analytics (machine learning for big data) as running examples.

Machine Learning Techniques

- Classification
- Regression
- Clustering
- Low-dimensional embeddings

Machine Learning

About 500-750 papers published at NIPS and ICML per year.

Motivation

Look at machine learning through the lens of optimization.

Motivation

Look at machine learning through the lens of optimization.

- Least squares regression
- SVMs
- Kernel learning
- k-means
- ...

seit 1558

Convex Optimization

LP	$\subseteq$	QP	$\subseteq$	SDP	$\subseteq$	convex Opt.	$\subseteq$	non-linear Opt.
lin. sep.		SVM		kernel learning		logistic regression		k-means
$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$

seit 1558

Convex Optimization

LP	$\subseteq$	QP	$\subseteq$	SDP	$\subseteq$	convex Opt.	$\subseteq$	non-linear Opt.
lin. sep.		SVM		kernel learning		logistic regression		k-means
$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$

A number of solvers / implementations for each problem.

Work flow

model problem

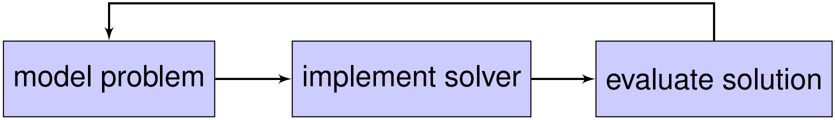
seit 1558

Work flow

model problem → implement solver

Work flow

Work flow



General Optimization in Machine Learning

Ideal world:

- One tool / algorithm for everything
- Easy to use
- As fast as hand-tuned specialized solvers

Non-Negative Least Squares Example

Problem:

$$\begin{aligned} \min_x \quad & \|Ax - b\|_2 \\ s.t. \quad & x \geq 0. \end{aligned}$$

Non-Negative Least Squares Example

Problem:

$$\begin{aligned} \min_x \quad & \|Ax - b\|_2 \\ s.t. \quad & x \geq 0. \end{aligned}$$

in CVX (modeling language in Matlab)

```
cvx_begin
  variable x(n)
  minimize(norm(A*x - b, 2))
  subject to
    x >= 0
cvx_end
```

seit 1558

Modeling Language and General Solvers

How does it work work?

Modeling Language and General Solvers

How does it work work?

CVX converts problem into standard form.

Modeling Language and General Solvers

How does it work work?

CVX converts problem into standard form.

General solver solves problem in standard form.

Modeling Language and General Solvers

How does it work work?

CVX converts problem into standard form.

General solver solves problem in standard form.

CVX converts solution back to original problem.

Modeling Language and General Solvers

Modeling Language: CVX

Developed at Stanford and Caltech (more than 10,000 software downloads per year)

Modeling Language and General Solvers

Modeling Language: CVX

Developed at Stanford and Caltech (more than 10,000 software downloads per year)

Solver SeDuMi:

Fastest general solver available (non-commercial)

Modeling Language and General Solvers

Modeling Language: CVX

Developed at Stanford and Caltech (more than 10,000 software downloads per year)

Solver SeDuMi:

Fastest general solver available (non-commercial)

Solver Gurobi:

Fastest general solver available (commercial)

Convex Optimization

LP	$\subseteq$	QP	$\subseteq$	SDP	$\subseteq$	convex Opt.	$\subseteq$	non-linear Opt.
lin. sep.		SVM		kernel learning		logistic regression		k-means
$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$

seit 1558

Convex Optimization

LP	$\subseteq$	QP	$\subseteq$	SDP	$\subseteq$	convex Opt.	$\subseteq$	non-linear Opt.
lin. sep.		**SVM**		kernel learning		logistic regression		k-means
$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$

Kernelized Dual SVM

Problem:

$$\begin{aligned} \min_\alpha \quad & 1/2\alpha^T K\alpha - \textstyle\sum_i \alpha_i \\ s.t. \quad & y^T\alpha = 0 \\ & 0 \leq \alpha \leq c. \end{aligned}$$

seit 1558

Kernelized Dual SVM

Problem:

$$\begin{aligned} \min_\alpha \quad & 1/2\alpha^T K\alpha - \textstyle\sum_i \alpha_i \\ s.t. \quad & y^T\alpha = 0 \\ & 0 \le \alpha \le c. \end{aligned}$$

in CVX

```
cvx_begin
  variable a(n)
  minimize(0.5*a'*K*a - sum(a))
  subject to
    y' * a == 0
    0 <= a <= c
cvx_end
```

Kernelized Dual SVM

	LIBSVM	CVX / SeDuMi	CVX / Gurobi
	Sec.	Sec.	Sec.
c	data set `a1a`		
1	0.28	376.6	57.8
4	0.30	392.1	44.6
c	data set `a7a`		
1	39.5	n/a	n/a
4	48.9	n/a	n/a

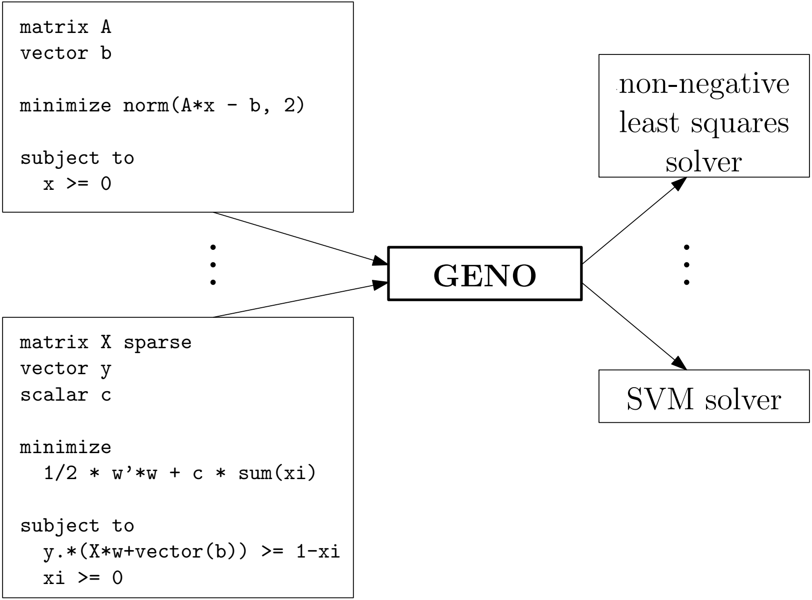
LIBSVM – Chang and Lin 2001
Cited more than 18,000 times.

seit 1558

GENO

Our approach

GENO



seit 1558

Kernelized Dual SVM

	LIBSVM Sec.	**GENO** Sec.	CVX / SeDuMi Sec.	CVX / Gurobi Sec.
c	data set `a1a`			
1	0.28	**0.28**	376.6	57.8
4	0.30	**0.33**	392.1	44.6
c	data set `a7a`			
1	39.5	**29.8**	n/a	n/a
4	48.9	**45.5**	n/a	n/a

LIBSVM – Chang and Lin 2001

seit 1558

Logistic Regression

	LIBLINEAR	**GENO**	CVX / SeDuMi	CVX / Gurobi
	Sec.	Sec.	Sec.	Sec.
c	data set `a1a`			
1	0.01	**0.02**	254.3	n/a
4	0.01	**0.03**	244.6	n/a
c	`rcv1_test (677,399; 47,236)`			
1	44.1	**36.3**	n/a	n/a
4	51.2	**52.1**	n/a	n/a

LIBLINEAR – Lin et al. (JMLR 2008)

Lasso

	glmnet naive	glmnet cov.	**GENO**	CVX/SeDuMi
	Sec.	Sec.	Sec.	Sec.
λ	data set $m = 800, n = 400$			
0.2	0.08	0.16	**0.29**	28.8
0.8	0.03	0.10	**0.25**	27.4
λ	data set $m = 8000, n = 4000$			
0.2	10.55	146.87	**23.42**	n/a
0.8	7.07	144.23	**27.50**	n/a

glmnet – Friedman, Hastie, Tibshirani (JStatSoft 2010)

Sparse PCA (non-linear version)

	GPower		**GENO**	
	fValue	Sec.	fValue	Sec.
λ	data set `colon-cancer`			
12.4	-72.3	0.0109	**-73.6**	**0.0083**
24.8	-36.7	0.0066	**-36.2**	**0.0171**
λ	data set `gisette_scale`			
0.1549	-34.5	1.98	**-34.6**	**1.52**
0.3098	-24.2	1.95	**-24.5**	**2.27**

GPower – Journée et al. (JMLR 2010)

Example – Demo

GENO

How does it work?

GENO

How does it work?

Tighter coupling of modeling language and solver.

GENO

How does it work?

Tighter coupling of modeling language and solver.

Combine thorough theoretical analysis with careful engineering.

Discussion

gradient computation

example: $f(x) = x^T A x$

Discussion

gradient computation

example: $f(x) = x^T A x$
gradient: $\nabla f(x) = (A^T + A)x$

Discussion

gradient computation

example: $f(x) = x^T A x$
gradient: $\nabla f(x) = (A^T + A)x$

example:
$f(w) = \frac{1}{2} w^T w + C \cdot \mathrm{sum}(\log(1 + \exp(-y. * (Xw + b))))$

seit 1558

Discussion

gradient computation

example: $f(x) = x^T A x$
gradient: $\nabla f(x) = (A^T + A)x$

example:
$f(w) = \frac{1}{2} w^T w + C \cdot \mathrm{sum}(\log(1 + \exp(-y. * (Xw + b))))$
gradient: $\nabla f(w) = \ ?$

Discussion

gradient computation

example: $f(x) = x^T A x$
gradient: $\nabla f(x) = (A^T + A)x$

example:
$f(w) = \frac{1}{2} w^T w + C \cdot \text{sum}(\log(1 + \exp(-y. * (Xw + b))))$
gradient: $\nabla f(w) = \ ?$

- Maple, Mathematica, Sage cannot do it

seit 1558

GENO

Results:

- One algorithm for ~~everything~~ a lot
- Easy to use
- Orders of magnitude faster than current general solvers
- As fast as hand-tuned specialized solvers

GENO

Results:

- One algorithm for ~~everything~~ a lot
- Easy to use
- Orders of magnitude faster than current general solvers
- As fast as hand-tuned specialized solvers
- Rapid prototyping and production quality code

SPP 1736 – "Algorithms for Big Data"

Scaling Up GENO for Big Data Analytics

1. Code for multi-core architectures

SPP 1736 – ”Algorithms for Big Data”

Scaling Up GENO for Big Data Analytics

1. Code for multi-core architectures
2. Code for GPGPUs

SPP 1736 – ”Algorithms for Big Data”

Scaling Up GENO for Big Data Analytics

1. Code for multi-core architectures
2. Code for GPGPUs
3. Code for MICs (Intel Xeon Phi)

seit 1558

SPP 1736 – "Algorithms for Big Data"

Scaling Up GENO for Big Data Analytics

1. Code for multi-core architectures
2. Code for GPGPUs
3. Code for MICs (Intel Xeon Phi)
4. Code for distributed solvers / cluster (on top of Spark)

Scaling up GENO

Two approaches

1. Parallelize basic linear algebra
2. Divide big problem into smaller problems; solve using ADMM

Very Preliminary Results

Parallelize linear algebra
Multi-core architecture (Intel Xeon 8-core)