PERMON

Parallel, Efficient, Robust, Modular, Object-oriented, Numerical computations

- quadratic programming solvers
- domain decomposition methods
- application-specific modules
- PETSc extension
- massively parallel
- HPC use

Martin Čermák, Václav Hapla*, David Horák, Jakub Kružík, Marek Pecha, Radim Sojka, Jiří Tomčala

Special thanks belong to Pavla Jirůtková, Alexandros Markopoulos, Lukáš Pospíšil, Pavel Skalný and Alena Vašatová.

http://permon.it4i.cz

*vaclav.hapla@vsb.cz

PermonSVM

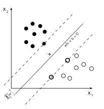
- supervised machine learning for binary classification problems
- solves dual soft-margin SVM problem
- designed for HPC platforms
- alternative to LIBLINEAR, Multi-core LIBLINEAR, MPI LIBLINEAR
- uses PermonQP during training procedure
  - SMALXE + solver for box constrained QP
- features
  - L1, L2 loss-functions
  - parser for LIBSVM data sets
  - grid search

```
PermonSVM svm;
PetscInt N_all, N_eq;
Mat Xt, Xt_test;
Vec y, y_test, y_out;

load_data(&Xt, &y, &Xt_test, &y_test);
VecDuplicate(y, &y_out);

PermonSVMCreate(comm, &svm);
PermonSVMSetTrainingSamples(svm, Xt, y);
PermonSVMSetFromOptions(svm);

PermonSVMTrain(svm);
PermonSVMClassify(svm, Xt_test, y_out);
PermonSVMTest(svm, Xt_test, y_test, &N_all, &N_eq);

PermonSVMDestroy(&svm); MatDestroy(&Xt); VecDestroy(&y);
MatDestroy(&Xt_test); VecDestroy(&y_test); VecDestroy(&y_out);
```

Soft-margin primal formulation

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i$$

$$\text{s.t.} \begin{cases} y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, \ \forall 1 \leq i \leq m, \\ \xi_i \geq 0. \ \forall 1 \leq i \leq m. \end{cases}$$



Soft-margin dual QP formulation

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m] \in \mathbb{R}^{n\times m}, \quad \mathbf{y} = [y_1, \ldots, y_m]^T,$$
$$\mathbf{e} = [1, 1, \ldots, 1] \in \mathbb{R}^m, \quad \mathbf{C} = [C, C, \ldots, C] \in \mathbb{R}^m,$$
$$\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]^T, \quad \mathbf{B_E} = [\mathbf{y}^T].$$

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{y}^T\mathbf{X}^T\mathbf{X}\mathbf{y}\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \text{ s.t. } \begin{cases} \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{C}, \\ \mathbf{B_E}\boldsymbol{\alpha} = 0. \end{cases}$$

Reconstruction formulas

Normal vector of hyperplane reconstruction formula

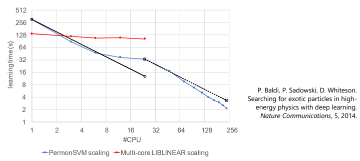
$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i y_i,$$

bias of hyperplane reconstruction formula

$$I^{SV} := \{i \mid \alpha_i > 0, \ i = 1, 2, \ \ldots, m\},$$
$$b = \frac{1}{|I^{SV}|}\sum_{i \in I^{sv}} \left(y_i - \mathbf{x}_i^T\mathbf{w}\right).$$

HIGGS dataset

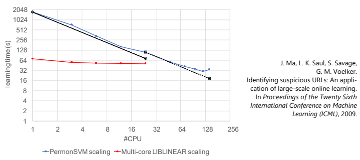
Training set contains **10.5M** data with **28** features, testing set contains **500k** data. Achieved accuracy: **64.37% (PermonSVM), 64.10% (Multi-core LIBLINEAR)**. Nonzero fill 100%.



P. Baldi, P. Sadowski, D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. Nature Communications, 5, 2014.

URL dataset

Training set contains **2.4M** data with **3.2M** features.
Achieved accuracy on training set: **98.51% (PermonSVM), 96.20% (Multi-core LIBLINEAR).** Nonzero fill 4%.



J. Ma, L. K. Saul, S. Savage, G. M. Voelker. Identifying suspicious URLs: An application of large-scale online learning. In Proceedings of the Twenty Sixth International Conference on Machine Learning (ICML), 2009.

computed on Salomon (IT4Innovations)

Ground truth supervised learning

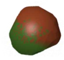
Detections of brittle and ductile fractures on DWTT surface of API L5 X-70 steel sheet

Workflow:

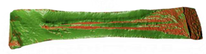
1. Perform Drop Weight-Tear Test, scan fracture surface as point cloud and create mesh

2. Expert determines brittle and ductile fracture ground truth

3. Extract normal vector characteristics of ground truth and train SVM



4. Use SVM model to detect types of fractures on whole surface

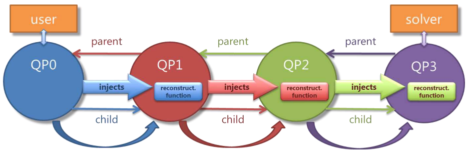


Ground truth training set contains **2148** data with **3** features. Achieved accuracy on training set: **93.25% (PermonSVM), 93.25% (Multi-core LIBLINEAR)**. Nonzero fill 100%. Training time: **0.02s (PermonSVM), 0.02s (Multi-core LIBLINEAR)**.

computed on MacBook Pro (Retina, 13-inch, Late 2012)

PermonQP

- framework for quadratic programming (QP)
- based on / extending PETSc
- QP problems, transforms, solvers
- easy-to-use / HPC-oriented
- workflow
  1. QP problem specification
  2. QP transforms
  3. automatic/manual choice of an appropriate solver



QP

```
QP parent, child; /* QP chain objects */
Vec x; /* solution */

Mat A; /* Hessian matrix */
Vec b; /* right-hand side */

Mat BE; Vec cE; /* linear equality constraints */
Vec lambdaE; /* equality Lagrange multipliers */

QPC constraint; /* inequality constraints */
```

$$\min_{\substack{l_b \leq x \leq u_b \\ B_{eq}x = c_{eq} \\ B_{ineq}x \leq c_{ineq}}} \frac{1}{2}x^TAx - x^Tb$$

QPS

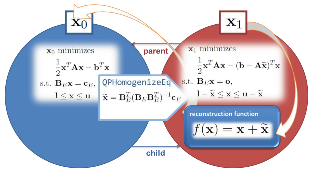
```
QP topQP, solQP; /* QP chain objects */
PetscReal rtol, atol, divtol; /* alg. tolerances */

PetscReal rnorm; PetscInt it; /* current state */
QPSMonitor monitor;
```

QPT

```
QPTDualize()
QPTEnforceEqByProjector()
QPTEnforceEqByPenalty()
QPTOrthonormalizeEq()
QPTHomogenizeEq()
QPTFetiPrepare()
```

Homogenization of equality constraints

- an example of QP transformation



Dualization

- crucial QP transform

$$\min \tfrac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u} - \mathbf{u}^T\mathbf{f} \text{ s.t. } \mathbf{B_E}\mathbf{u} = \mathbf{o} \text{ and } \mathbf{B_I}\mathbf{u} \leq \mathbf{c_I}$$
$$\min \tfrac{1}{2}\boldsymbol{\lambda}^T\mathbf{A}\boldsymbol{\lambda} - \boldsymbol{\lambda}^T\mathbf{b} \text{ s.t. } \mathbf{C}\boldsymbol{\lambda} = \mathbf{d} \text{ and } \boldsymbol{\lambda} \geq \boldsymbol{l}$$

- new QP with smaller dimensions, better conditioned and simpler constraints

SMALXE

$$\min \tfrac{1}{2}\boldsymbol{\lambda}^T\mathbf{A}\boldsymbol{\lambda} - \boldsymbol{\lambda}^T\mathbf{b} \text{ s.t. } \mathbf{C}\boldsymbol{\lambda} = \mathbf{d} \text{ and } \boldsymbol{\lambda} \geq \boldsymbol{l}$$

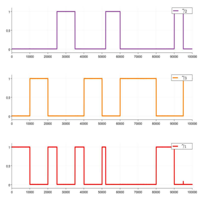
- "pass-through" solver taking care of the equality constraints
- Hessian matrix augmented with the penalty term
- an auxiliary problem with the rest of the constraints is solved by the inner solver

Specific solvers pluggable into SMALXE

- unconstrained $\min \tfrac{1}{2}\boldsymbol{\lambda}^T\mathbf{A}\boldsymbol{\lambda} - \boldsymbol{\lambda}^T\mathbf{b} \text{ s.t. } \boldsymbol{\lambda} \geq \boldsymbol{l}$
  - PETSc KSP and TAO wrapper
- box constraints
  - MPRGP
  - PETSc TAO wrapper (GPCG, BLMVM, TRON)
- separable convex constraints
  - MPGP, PBBf, SPG-QP

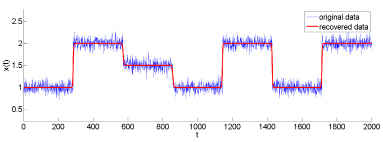
Nonparametric big data time series de-noising, modelling & clustering

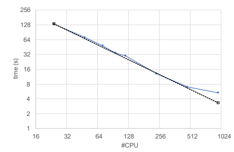
- PERMON is used as inner QP solver in new open-source HPC library for nonstationary time-series analysis in C++ developed by group of Illia Horenko (USI Lugano)
- popular FEM-H1 methodology is based on minimization of regularized averaged clustering functional with respect to linear equality and box constraints
- the method identifies the locally stationary models on clusters and the parameters of these models
- the size of the problem is given by the number of clusters multiplied with the length of given time series
- long time series cannot be operated in one node; the PETSc parallel vectors and regularization matrices come into play



$$L_\varepsilon(\Theta, \Gamma) = \sum_{t=1}^{T}\sum_{i=1}^{K}\gamma_i(t)g(x_t, \Theta_i) + \varepsilon^2\sum_{i=1}^{K}\sum_{t=1}^{T}\left(\gamma_i(t-1) - \gamma_i(t)\right)^2$$
$$\sum_{i=1}^{K}\gamma_i(t) = 1, \forall t, \ 0 \leq \gamma_i(t) \leq 1, \forall t, i.$$

scalability test ⇩: 1D signal of length T=1e7 with additive Gaussian noise with variance sigma=0.2 was modelled by FEM-H1 on K=3 clusters with optimal penalty parameter epsilon^2=3





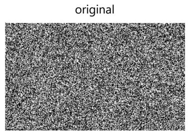
computed on Salomon (IT4Innovations)

- regularization can be implemented both in time and space
- example ⇨: in the case of image de-noising, the graph of regular 2D grid is used

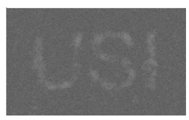
Università della Svizzera italiana

application by Lukáš Pospíšil, USI
https://github.com/eth-cscs/PASC_inference

original



recovered



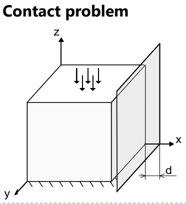
PermonFLLOP

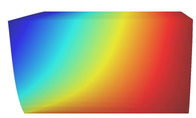
- extends PermonQP with domain decomposition methods of the FETI type
- problems with or without inequality constraints
- scalability up to tens of thousands of CPU cores, billions of unknowns
- assembly of FETI-specific objects

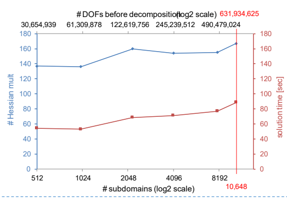
```
Mat Ks, BIs; Vec fs, cI, coords;
IS l2g, dbcis; MPI_Comm comm;
FLLOP fllop;

/*  generate the data */

/*  create FLLOP living in communicator */
FllopCreate(comm, &fllop);

/*  set the subdomain stiffness matrix and load vector */
FllopSetStiffnessMatrix(fllop, Ks);
FllopSetLoadVector(fllop, fs);

/*  set the local-to-global mapping for gluing */
FllopSetLocalToGlobalMapping(fllop, l2g);

/*  specify the Dirichlet conditions in the local numbering and tell FLLOP
to enforce them by means of the B matrix */
FllopAddDirichlet(fllop, dbcis, FETI_LOCAL, FETI_DBC_B);

/*  set vertex coordinates for rigid body modes */
FllopSetCoordinates(fllop, coords);

/*  set the non-penetration inequality constraints */
FllopSetIneq(fllop, BIs, cI);

FllopSolve(fllop);
```

```
/* FllopSolve() function */

/*  subdomain data */
Mat Ks, BIs, Bgs, Bds, Rs;  Vec fs;
/* global data */
Mat K, BI, Bg, Bd, R;  Vec f, cI, cd;
/*  QP problem, QP solvers */
QP qp; QPS qps;

/*  create a QP data structure */
QPCreate(comm,&qp);

/*  Globalise the data. */
MatCreateBlockDiag(Ks,&K);
MatCreateBlockDiag(Rs,&R);
MatMerge(Bgs,&Bg); MatMerge(Bds,&Bd);
MatMerge(BIs,&BI); VecMerge(fs,&f);

/*  Insert the data into the QP. */
QPSetOperator(qp, K);
QPSetOperatorNullspace(qp, R);
QPSetRHS(qp, f);
QPAddEq(qp, Bg, NULL);
QPAddEq(qp, Bd, cd);
QPSetIneq(qp, BI, cI);

/*  Basic sequence of QP transforms
    giving (T)FETI method.
    QP chain is created in backend. */
QPTDualize(qp);
QPTHomogenizeEq(qp);
QPTEnforceEqByProjector(qp);

/*  Create a PermonQP solver. */
QPSCreate(comm, &qps);

/* Set the QP to be solved. */
QPSSetQP(qps,qp);

/*  Solve, i.e. hand over to PermonQP.
    The last QP in the chain is solved. */
QPSSolve(qps);
```

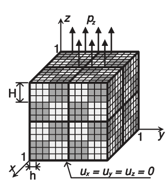
Contact problem

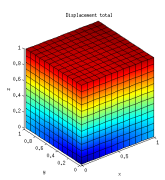


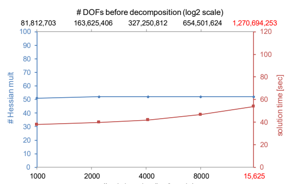




Linear elasticity problem







computed on Archer (EPCC)

IT4Innovations national supercomputing center
Department of Applied Mathematics