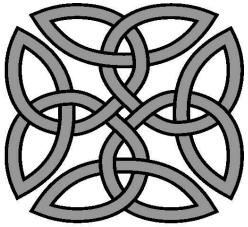


School of Mathematics

Using interior point methods for optimization in training very large scale Support Vector Machines

Jacek Gondzio
Email: J.Gondzio@ed.ac.uk
URL: http://www.maths.ed.ac.uk/~gondzio

joint work with **Kristian Woodsend**

Outline

- **Interior Point Methods for QP**
  - logarithmic barrier function
  - complementarity conditions
  - linear algebra
- **Support Vector Machine training**
  - Quadratic Programming formulation
  - specific features
- **IPMs for SVM training**
  - separable formulations !!!
  - linear and nonlinear kernels in SVMs
- **Challenges remaining**
  - nonlinear kernels in SVMs
  - indefinite kernels in SVMs
- **Conclusions**

Part 1:

Interior Point Methods for QP

"Elements" of the IPM

What do we need to derive the **Interior Point Method**?

- logarithmic barriers.
- duality theory:
  Lagrangian function;
  first order optimality conditions.
- Newton method.

Wright, Primal-Dual Interior-Point Methods, SIAM, 1997.

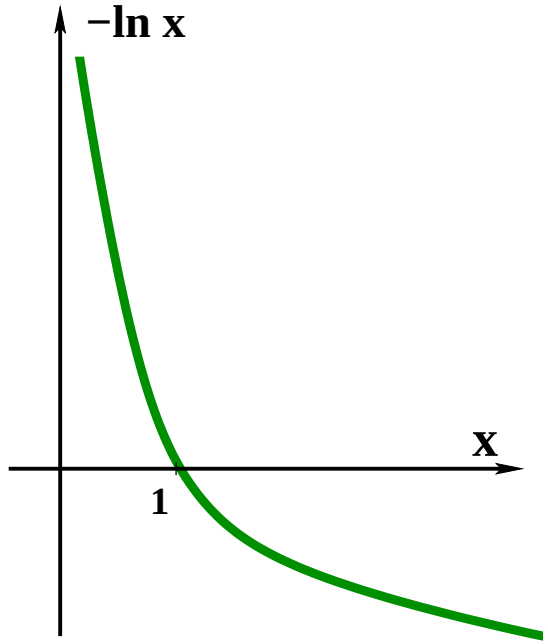
Andersen, Gondzio, Mészáros and Xu,
Implementation of Interior Point Methods for Large Scale Linear Programming, in: *Interior Point Methods in Mathematical Programming*, T Terlaky (ed.), Kluwer Academic, 1996, pp. 189–252.

Logarithmic barrier

$-\ln v_i$

"replaces" the inequality

$v_i \geq 0$.



Observe that

$$\min \mathrm{e}^{-\sum_{i=1}^{n} \ln v_i} \iff \max \prod_{i=1}^{n} v_i$$

The minimization of $-\sum_{i=1}^{n} \ln v_i$ is equivalent to the maximization of the product of distances from all hyperplanes defining the positive orthant: it prevents all v_i from approaching zero.

Logarithmic barrier

Replace the **primal** QP

$$\begin{array}{ll} \min & c^T v + \frac{1}{2} v^T Q v \\ \text{s.t.} & Av = b, \\ & v \geq 0, \end{array}$$

with the **primal barrier program**

$$\begin{array}{ll} \min & c^T v + \frac{1}{2} v^T Q v - \mu \sum_{j=1}^{n} \ln v_j \\ \text{s.t.} & Av = b. \end{array}$$

Lagrangian:

$$L(v, \lambda, \mu) = c^T v + \frac{1}{2} v^T Q v - \lambda^T (Av - b) - \mu \sum_{j=1}^{n} \ln v_j.$$

Conditions for a stationary point of the Lagrangian

$$\begin{aligned} \nabla_v L(v,\lambda,\mu) &= c - A^T\lambda + Qv - \mu V^{-1}e = 0 \\ \nabla_\lambda L(v,\lambda,\mu) &= Av - b = 0, \end{aligned}$$

where $V^{-1} = diag\{v_1^{-1}, v_2^{-1}, \cdots, v_n^{-1}\}$.

Let us denote

$$s = \mu V^{-1}e, \quad \text{i.e.} \quad VSe = \mu e.$$

The **First Order Optimality Conditions** are:

$$\begin{aligned} Av &= b, \\ A^T\lambda + s - Qv &= c, \\ VSe &= \mu e, \\ (v,s) &> 0. \end{aligned}$$

First Order Optimality Conditions

Active-set Method:

$$\begin{aligned} Av &= b \\ A^T\lambda + s - Qv &= c \\ VSe &= 0 \\ v,\, s &\geq 0. \end{aligned}$$

Interior Point Method:

$$\begin{aligned} Av &= b \\ A^T\lambda + s - Qv &= c \\ VSe &= \mu e \\ v,\, s &\geq 0. \end{aligned}$$

Complementarity $v_i \cdot s_i = 0 \quad \forall i = 1, 2, ..., n.$

Active-set Method makes a guess of optimal partition:
$\mathcal{A} \cup \mathcal{I} = \{1, 2, ..., n\}$.

For *active* constraints ($i \in \mathcal{A}$), $v_i = 0$ and

$$v_i \cdot s_i = 0 \quad \forall i \in \mathcal{A}.$$

For *inactive* constraints ($i \in \mathcal{I}$), $s_i = 0$ hence

$$v_i \cdot s_i = 0 \quad \forall i \in \mathcal{I}.$$

Interior Point Method uses ε-mathematics:

Replace $\quad v_i \cdot s_i = 0 \quad \forall i = 1, 2, ..., n$
by $\quad v_i \cdot s_i = \mu \quad \forall i = 1, 2, ..., n.$

Force convergence $\mu \to 0$.

Apply Newton Method to the FOC

The first order optimality conditions for the barrier problem form a large system of nonlinear equations

$$f(v, \lambda, s) = 0,$$

where $f : \mathcal{R}^{2n+m} \mapsto \mathcal{R}^{2n+m}$ is a mapping defined as follows:

$$f(v, \lambda, s) = \begin{bmatrix} Av - b \\ A^T\lambda + s - Qv - c \\ VSe - \mu e \end{bmatrix}.$$

Actually, the first two terms of it are **linear**; only the last one, corresponding to the complementarity condition, is **nonlinear**.

Newton Method (cont'd)

Note that

$$\nabla f(v,\lambda,s) = \begin{bmatrix} A & 0 & 0 \\ -Q & A^T & I \\ S & 0 & V \end{bmatrix}.$$

Thus, for a given point (v,λ,s) we find the Newton direction $(\Delta v, \Delta\lambda, \Delta s)$ by solving the system of linear equations:

$$\begin{bmatrix} A & 0 & 0 \\ -Q & A^T & I \\ S & 0 & V \end{bmatrix} \cdot \begin{bmatrix} \Delta v \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} b - Av \\ c - A^T\lambda - s + Qv \\ \mu e - VSe \end{bmatrix}.$$

Linear Algebra of IPM for QP

$$\begin{bmatrix} A & 0 & 0 \\ -Q & A^T & I \\ S & 0 & V \end{bmatrix} \begin{bmatrix} \Delta v \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} \xi_p \\ \xi_d \\ \xi_\mu \end{bmatrix} = \begin{bmatrix} b - Av \\ c - A^T\lambda - s + Qv \\ \mu e - VSe \end{bmatrix}.$$

Use the third equation to eliminate

$$\begin{aligned} \Delta s &= V^{-1}(\xi_\mu - S\Delta v) \\ &= -V^{-1}S\Delta v + V^{-1}\xi_\mu, \end{aligned}$$

from the second equation and get

$$\begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta v \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} \xi_d - V^{-1}\xi_\mu \\ \xi_p \end{bmatrix}.$$

where $\Theta = VS^{-1}$ is a diagonal scaling matrix.

Θ is always very **ill-conditioned**.

Augmented system

$$\begin{bmatrix} -Q-\Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta v \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} r \\ h \end{bmatrix} = \begin{bmatrix} \xi_d - V^{-1}\xi_\mu \\ \xi_p \end{bmatrix}.$$

Symmetric but indefinite linear system.
In general, it may be difficult to solve.

Separable Quadratic Programs

When matrix Q is diagonal ($Q = D$), the augmented system can be further reduced. Eliminate

$$\Delta v = (D + \Theta^{-1})^{-1}(A^T\Delta\lambda - r),$$

to get **normal equations** (symmetric, positive definite system)

$$(A(D + \Theta^{-1})^{-1}A^T)\Delta\lambda = g = A(D + \Theta^{-1})^{-1}r + h.$$

Sparsity Issues in QP

Observation: the inverse of the sparse matrix may be dense.

Example

$$\begin{bmatrix} 1 & 1 & & & \\ 1 & 2 & 1 & & \\ & 1 & 2 & 1 & \\ & & 1 & 2 & 1 \\ & & & 1 & 2 \end{bmatrix}^{-1} = \left(\begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ & 1 & 1 & & \\ & & 1 & 1 & \\ & & & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & 1 & 1 & \\ & & & 1 & 1 \\ & & & & 1 \end{bmatrix} \right)^{-1}$$

$$= \begin{bmatrix} 1 & -1 & 1 & -1 & 1 \\ & 1 & -1 & 1 & -1 \\ & & 1 & -1 & 1 \\ & & & 1 & -1 \\ & & & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ 1 & -1 & 1 & & \\ -1 & 1 & -1 & 1 & \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & -4 & 3 & -2 & \\ -4 & 4 & -3 & 2 & \\ 3 & -3 & 3 & -2 & \\ -2 & 2 & -2 & 2 & \\ 1 & -1 & 1 & -1 & \end{bmatrix}$$

IPMs for QP:
Do not explicitly invert the matrix $Q + \Theta^{-1}$
to form $A(Q + \Theta^{-1})^{-1}A^T$ unless Q is **diagonal**.

Interior Point Methods

Theory: IPMs converge in $\mathcal{O}(\sqrt{n})$ or $\mathcal{O}(n)$ iterations
Practice: IPMs converge in $\mathcal{O}(\log n)$ iterations
... but one iteration may be expensive!

Suppose $A \in \mathcal{R}^{m \times n}$ is a dense matrix.
Major computational effort when solving separable QP
(separable QP means that $Q = D$, diagonal).

build $H = A(Q + \Theta^{-1})^{-1}A^T$	$\mathcal{O}(nm^2)$
compute Cholesky $H = L\Lambda L^T$	$\mathcal{O}(m^3)$

Recall $n \gg m$.

Ill-conditioning of $\Theta = VS^{-1}$

For **active** constraints:	$\Theta_j = v_j/s_j \to 0$	$\Theta_j^{-1} \to \infty$;
For **inactive** constraints:	$\Theta_j = v_j/s_j \to \infty$	$\Theta_j^{-1} \to 0$.

Goldfarb and Scheinberg, A product form Cholesky factorization for handling dense columns in IPMs for linear programming, *Mathematical Programming*, 99(2004) 1-34.

Although $\tilde{\Theta} = (Q + \Theta^{-1})^{-1}$ behaves badly, the Cholesky factorization $H = L\Lambda L^T$ behaves well:

Λ captures instability (variability) of $\tilde{\Theta}$;
L is well conditioned (bounded independently of $\tilde{\Theta}$).

Represent $L = L_1 L_2 \dots L_m$, where L_i has entries only in column i.

Drawback: PFCF is **sequential** by nature.

Interior Point Methods: Summary

- **Interior Point Methods for QP**
  - polynomial algorithms
  - excellent practical behaviour
  - competitive for small problems ($\leq$ 1,000,000 vars)
  - beyond competition for large problems ($\geq$ 1,000,000 vars)

- **Opportunities for SVM training with IPMs**
  - dense data
  - very large size
  - well-suited to parallelism

Part 2:

Support Vector Machine training

Classification

We consider a set of points $X = \{x_1, x_2, \ldots, x_n\}$, $x_i \in \mathcal{R}^m$ to be classified into two subsest of "good" and "bad" ones.

$X = G \cup B$ and $G \cap B = \emptyset$.

We look for a function $f : X \mapsto \mathcal{R}$ such that
$f(x) \geq 0$ if $x \in G$ and
$f(x) < 0$ if $x \in B$.

Usually $n \gg m$.

Linear Classification

We consider a case when f is a linear function:

$$f(x) = w^T x + b,$$

where $w \in \mathcal{R}^m$ and $b \in \mathcal{R}$.
In other words we look for a hyperplane which separates "good" points from "bad" ones.

In such case the *decision rule* is given by $y = sgn(f(x))$.
If $f(x_i) \geq 0$, then $y_i = +1$ and $x_i \in G$.
If $f(x_i) < 0$, then $y_i = -1$ and $x_i \in B$.

We say that there is a linearly separable training sample

$$S = ((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)).$$

How does it work?

Given a linearly separable database (training sample)

$$S = ((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n))$$

find a separating hyperplane

$$w^T x + b = 0,$$

which satisfies

$$y_i(w^T x_i + b) \geq 1, \quad \forall i = 1, 2, \ldots, n.$$

Given a new (unclassified) point x_0, compute

$$y_0 = sgn(w^T x_0 + b)$$

to decide whether x_0 is "good" or "bad".

Separating Hyperplane

To guarantee a nonzero margin of separation we look for a hyperplane

$$w^T x + b = 0,$$

such that

$w^T x_i + b \geq 1$ for "good" points; $w^T x_i + b \leq -1$ for "bad" points.

This is equivalent to:

$$\frac{w^T x_i}{\|w\|} + \frac{b}{\|w\|} \geq \frac{1}{\|w\|} \quad \text{for "good" points;}$$

$$\frac{w^T x_i}{\|w\|} + \frac{b}{\|w\|} \leq -\frac{1}{\|w\|} \quad \text{for "bad" points.}$$

In this formualtion the normal vector of the separating hyperplane $\frac{w}{\|w\|}$ has unit length. In this case the margin between "good" and "bad" points is measured by $\frac{2}{\|w\|}$. This margin should be maximised.

This can be achieved by **minimising** the norm $\|w\|$.

QP Formulation

Finding a separating hyperplane can be formulated as a quadratic programming problem:

$$\begin{aligned} \min \quad & \frac{1}{2} w^T w \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1, \quad \forall i = 1, 2, \ldots, n. \end{aligned}$$

In this formulation the Euclidean norm of w is minimized.
This is clearly a convex optimization problem.

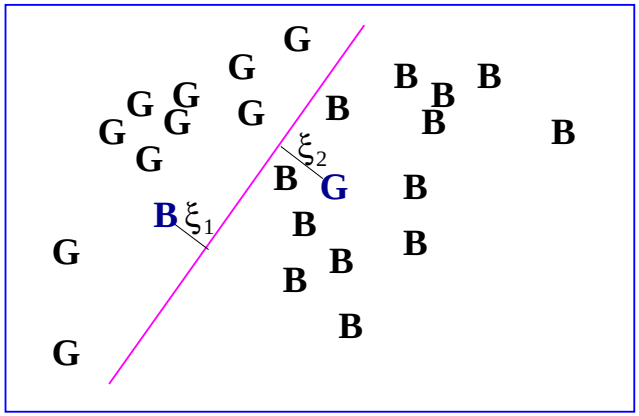
(We can minimize $\|w\|_1$ or $\|w\|_\infty$ and then the problem can be reformulated as an LP.)

Two major difficulties:

- Clusters may not be separable at all
  $\longrightarrow$ minimize the error of misclassifications;
- Clusters may be separable by a nonlinear manifold
  $\longrightarrow$ find the right *feature map*.

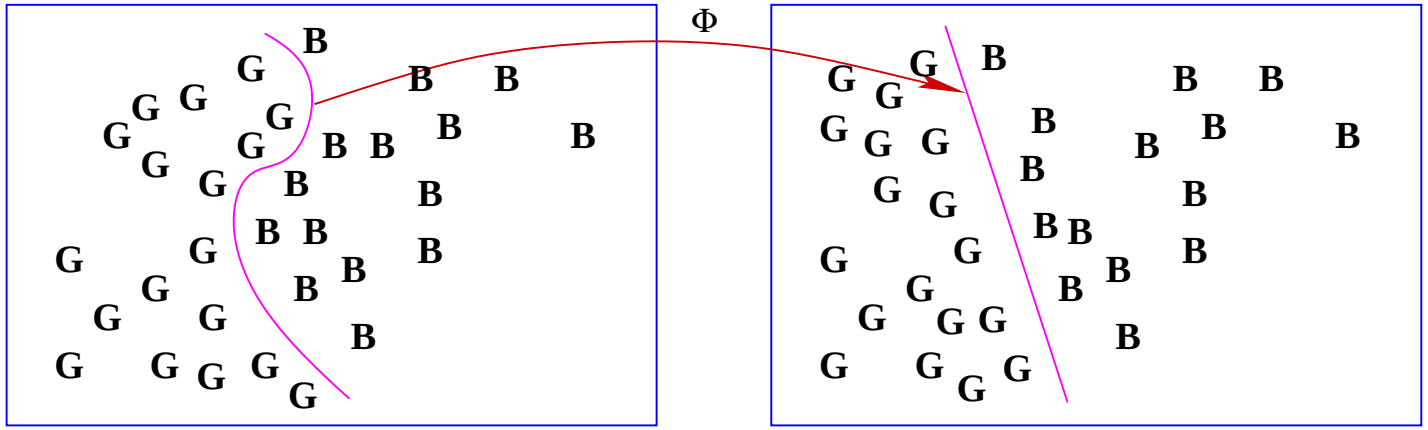
Difficult Cases

Nonseparable clusters:



Errors when defining clusters of good and bad points.
Minimize the global error of misclassifications: $\xi_1 + \xi_2$.

Use nonlinear feature map:



Linearly nonseparable case

If perfect linear separation is impossible then for each misclassified data we introduce a slack variable ξ_i which measures the distance between the hyperplane and misclassified data.

Finding the best hyperplane can be formulated as a quadratic programming problem:

$$\begin{aligned} \min \quad & \tfrac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & y_i (w^T x_i + b) + \xi_i \geq 1, \qquad \forall i = 1, 2, \ldots, n, \\ & \xi_i \geq 0 \qquad\qquad\qquad\quad \forall i = 1, 2, \ldots, n, \end{aligned}$$

where C $(C > 0)$ controls the penalisation for misclassifications.

We will derive the **dual** quadratic problem.

We associate Lagrange multipliers $z \in \mathcal{R}^n\,(z{\geq}0)$ and $s \in \mathcal{R}^n\,(s{\geq}0)$ with the constraints $y_i(w^T x_i + b) + \xi_i \geq 1$ and $\xi \geq 0$, and write the **Lagrangian**

$$L(w, b, \xi, z, s) = \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} z_i (y_i (w^T x_i + b) + \xi_i - 1) - s^T \xi.$$

SVM community uses α instead of z.

Dual Quadratic Problem

Stationarity conditions (with respect to all primal variables):

$$\begin{aligned}
\nabla_w L(w,b,\xi,z,s) &= w - \sum_{i=1}^{n} y_i x_i z_i &= 0 \\
\nabla_{\xi_i} L(w,b,\xi,z,s) &= C - z_i - s_i &= 0 \\
\nabla_b L(w,b,\xi,z,s) &= \sum_{i=1}^{n} y_i z_i &= 0.
\end{aligned}$$

Substituting these equations into the Lagrangian function we get

$$L(w,b,\xi,z,s) = \sum_{i=1}^{n} z_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j (x_i^T x_j) z_i z_j.$$

Hence the **dual** problem has the form:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} z_i - \tfrac{1}{2} \sum_{i,j=1}^{n} y_i y_j (x_i^T x_j) z_i z_j \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_i z_i = 0, \\ & 0 \le z_i \le C, \qquad \forall i = 1, 2, \ldots, n, \end{aligned}$$

SVM community uses α instead of z.

Dual Quadratic Problem (continued)

Observe that the dual problem has a neat formulation in which only dual variables z are present. (The primal variables (w, b, ξ) do not appear in the dual.)

Define a **dense** matrix $Q \in \mathcal{R}^{n \times n}$ such that $q_{ij} = y_i y_j (x_i^T x_j)$.

Rewrite the (dual) quadratic program:

$$\begin{aligned} \max \quad & e^T z - \tfrac{1}{2} z^T Q z, \\ \text{s.t.} \quad & y^T z = 0, \\ & 0 \leq z \leq Ce, \end{aligned}$$

where e is the vector of ones in $\mathcal{R}^n$.

The matrix Q corresponds to a specific **linear kernel function**.

Dual Quadratic Problem (continued)

The primal problem is convex hence the dual problem must be well defined too. The dual problem is to maximise the concave function. We can prove it directly.

Lemma: The matrix Q is positive semidefinite.

Proof: Define

$$G = [y_1 x_1 | y_2 x_2 | \ldots | y_n x_n]^T \in \mathcal{R}^{n \times m}$$

and observe that

$$Q = GG^T \quad \text{(i.e.,} \quad q_{ij} = y_i y_j (x_i^T x_j)).$$

For any $z \in \mathcal{R}^n$ we have

$$z^T Q z = (z^T G)(G^T z) = \|G^T z\|^2 \geq 0$$

hence Q is positive semidefinite.

Part 3:

IPMs for Support Vector Machine training

Interior Point Methods in SVM Context

Fine and Scheinberg, Efficient SVM training using low-rank kernel representations, *J. of Machine Learning Res.*, 2(2002) 243-264.

Ferris and Munson, Interior point methods for massive support vector machines, *SIAM J. on Optimization*, 13(2003) 783-804.

Woodsend and Gondzio, Exploiting separability in large-scale linear SVM training, Tech Rep MS-07-002, Edinburgh 2007.
`http://www.maths.ed.ac.uk/~gondzio/reports/wgSVM.html`

Unified framework which includes:

- Classification (ℓ_1 and ℓ_2 error)
- Universum SVM
- Ordinal Regression
- Regression

Reformulate QPs as **separable**.

IPMs for SVMs: Exploit separability

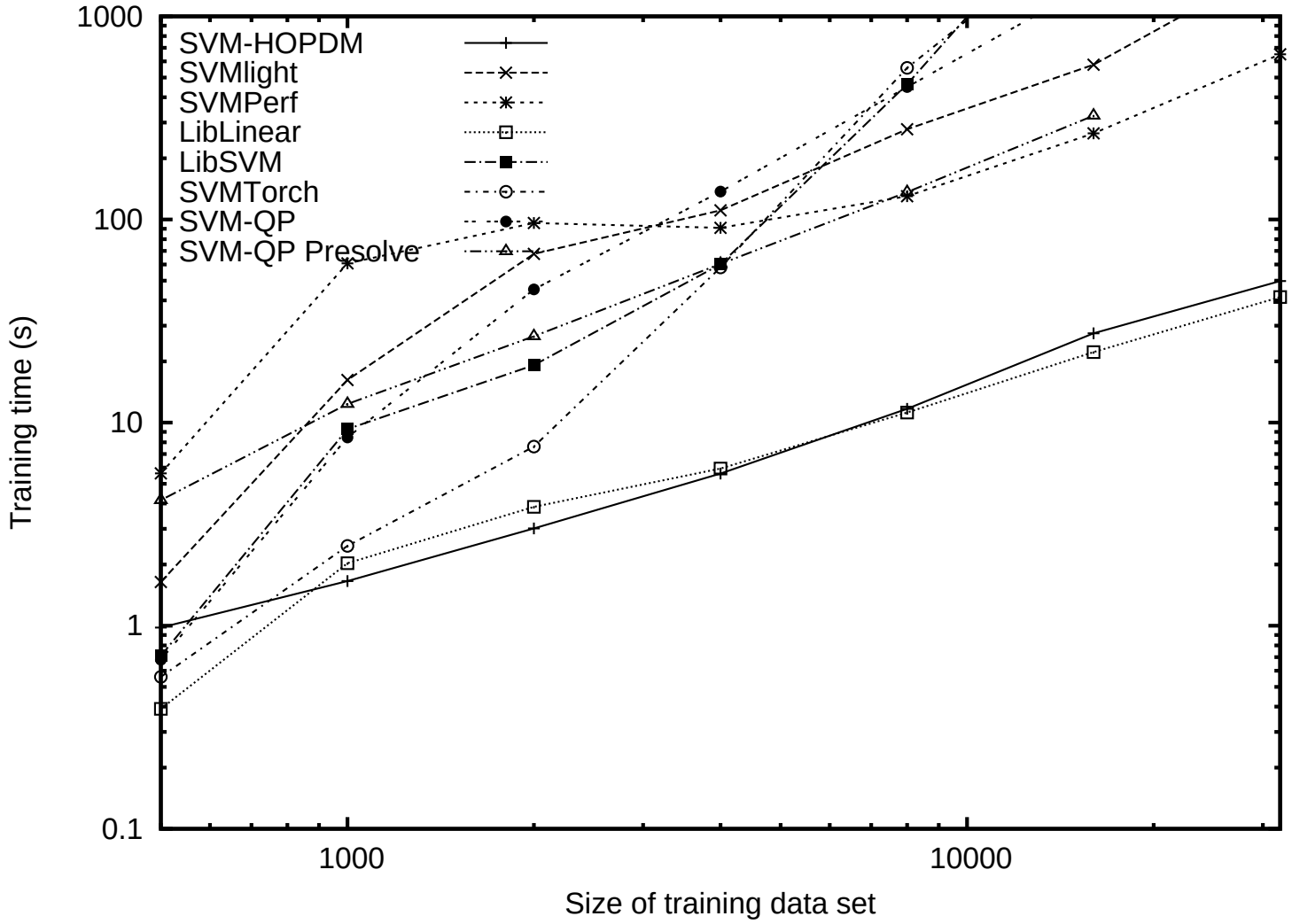
Key trick: represent $Q = F^T DF$, where $F \in \mathcal{R}^{k \times n}$, $k \ll n$.
Introduce new variable $u = Fz$.

Observe: $z^T Q z = z^T F^T DFz = u^T Du$.

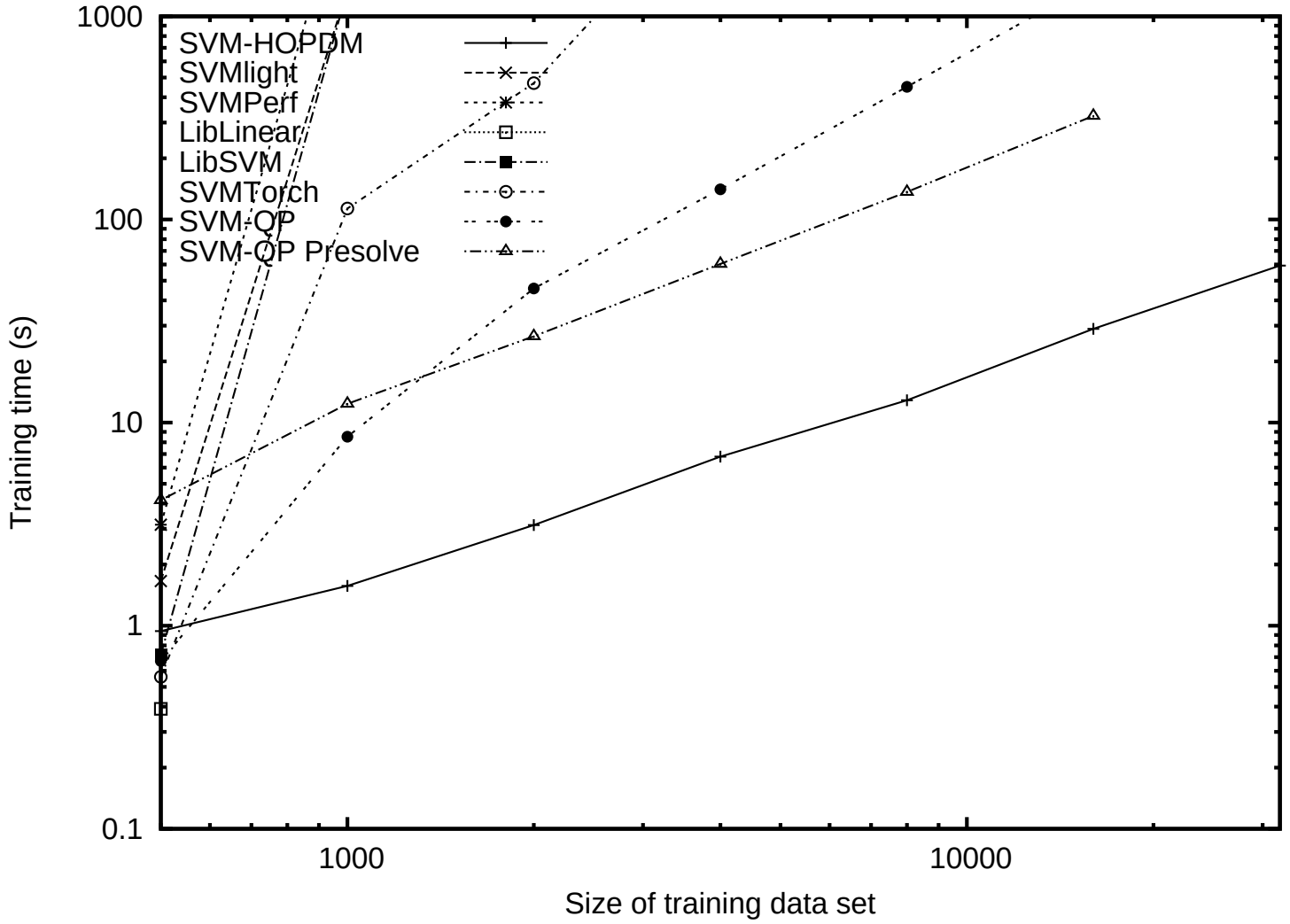
$$\begin{array}{ll} \min & c^T z + \frac{1}{2} z^T Q z \\ \text{s.t.} & Az = b, \\ & z \geq 0. \end{array} \quad \Longleftrightarrow \quad \begin{array}{ll} \min & c^T z + \frac{1}{2} u^T Du \\ \text{s.t.} & Az = b, \\ & Fz - u = 0, \\ & z \geq 0. \end{array}$$

non-separable QP	separable QP
m constraints	$m + k$ constraints
n variables	$n + k$ variables

Comparison: SVM-HOPDM vs other algorithms
Data with 255 attributes. $C = 1$, 10% misclassified.



Comparison: SVM-HOPDM vs other algorithms
Data with 255 attributes. $C = 100$, 10% misclassified.



Comparison: of training times using real-world data sets.
Each data set was trained using $C = 1, 10$ and 100.
NC indicates that the method did not converge to a solution.

Data set ($n \times m$)	C	SVM-HOPDM	SVM^{light}	SVM^{perf}	LIB-LINEAR	LIBSVM	SVMTorch	SVM-QP	SVM-Q presolv
Adult	1	16.5	87.7	280.7	1.6	192.4	621.8	164.5	188.
32561×123	10	26.5	1043.3	3628.0	9.3	857.7	5046.0	284.1	206.
	100	27.9	10447.4	29147.2	64.2	5572.1	44962.5	544.8	216.
Covtype	1	47.7	992.4	795.6	8.5	2085.8	2187.9	731.8	405.
150000×54	10	52.7	6021.2	12274.5	34.3	2516.7	10880.6	971.6	441.
	100	55.4	66263.8	58699.8	235.2	6588.0	74418.1	1581.8	457.
MNIST	1	79.6	262.9	754.1	9.3	197.1	660.1	233.0	1019.
10000×780	10	83.4	3425.5	8286.8	65.4	1275.2	5748.1	349.4	1104.
	100	86.2	NC	196789.0	NC	11456.4	54360.6	602.5	1267.
SensIT	1	55.2	913.5	8418.3	53.6	2542.0	2814.4	535.2	456.
78823×100	10	60.1	7797.4	> 125000	369.1	7867.8	21127.8	875.4	470.
	100	63.6	NC	> 125000	NC	49293.7	204642.6	1650.1	489.
USPS	1	13.2	15.0	40.9	4.4	10.4	7.7	51.2	117.
7291×256	10	14.2	147.4	346.6	27.7	20.9	23.9	64.7	127.
	100	14.3	1345.2	2079.5	NC	93.8	142.4	86.9	143.

Computational effort of IPM-based SVM implementation:

$$\text{build } H = A(Q + \Theta^{-1})^{-1}A^T \qquad \mathcal{O}(nm^2)$$

$$\text{compute Cholesky } H = L\Lambda L^T \qquad \mathcal{O}(m^3)$$

Attempts to reduce this effort

Gertz and Griffin, SVM classifiers for large datasets, Tech Rep ANL/MCS-TM-289, Argonne National Lab, 2005.

$\longrightarrow$ use iterative method (preconditioned conjugate gradient).

Jung, O'Leary and Tits, Adaptive constraint reduction for training SVMs, *Electronic Trans on Num Analysis* 31(2008) 156-177.

$\longrightarrow$ use a subset of points $n_1 \ll n$
mimic "active-set" strategy within IPM.

Parallelism

Exploit **bordered block-diagonal structure** in augm. system
Break H into blocks:

$$H = \begin{bmatrix} H_1 & & & & A_1^T \\ & H_2 & & & A_2^T \\ & & \ddots & & \vdots \\ & & & H_p & A_p^T \\ A_1 & A_2 & \dots & A_p & 0 \end{bmatrix},$$

and decompose

$$H = \begin{bmatrix} L_1 & & & \\ & \ddots & & \\ & & L_p & \\ L_{A_1} & \dots & L_{A_p} & L_0 \end{bmatrix} \begin{bmatrix} \Lambda_1 & & & \\ & \ddots & & \\ & & \Lambda_p & \\ & & & \Lambda_0 \end{bmatrix} \begin{bmatrix} L_1^T & & & L_{A_1}^T \\ & \ddots & & \vdots \\ & & L_p^T & L_{A_p}^T \\ & & & L_0^T \end{bmatrix}$$

Parallelism (continued)

- Cholesky factor preserves block-structure:

$$\begin{aligned} H_i &= L_i \Lambda_i L_i^T, \quad L_i = I, \, \Lambda_i = H_i, \quad i = 1..p \\ L_{A_i} &= A_i L_i^{-T} \Lambda_i^{-1} = A_i H_i^{-1}, \quad i = 1..p \\ H_0 &= -\sum_{i=1}^{p} A_i H_i^{-1} A_i^T = L_0 \Lambda_0 L_0^T \end{aligned}$$

- And the system $H \begin{bmatrix} \Delta v \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} r \\ h \end{bmatrix}$ is solved by

$$\begin{aligned} t_i &= L_i^{-1} r_i, \; i = 1..p \\ t_0 &= L_0^{-1} (h - \sum L_{A_i} t_i) \\ q_i &= \Lambda_i^{-1} t_i, \; i = 0..p \\ \Delta\lambda &= L_0^{-T} q_0 \\ \Delta v_i &= L_i^{-T} (q_i - L_{A_i}^T \Delta\lambda), \quad i = 1..p \end{aligned}$$

- Operations (Cholesky, Solve, Product) performed on sub-blocks

Comparison: Parallel software

PASCAL Large Scale Learning Challenge

`http://largescale.first.fraunhofer.de/about/`

Data set	n	m
Alpha	500000	500
Beta	500000	500
Gamma	500000	500
Delta	500000	500
Espilon	500000	2000
Zeta	500000	2000
FD	2560000	900
OCR	3500000	1156
DNA	6000000	800

OOPS Object-Oriented Parallel Solver

`http://www.maths.ed.ac.uk/~gondzio/parallel/solver.html`

Dataset	# cores	C	OOPS	PGPDT	PSVM	Milde
Alpha	16	1	39	3673	1684	(80611)
		0.01	50	4269	4824	(85120)
Beta	16	1	120	5003	2390	(83407)
		0.01	48	4738	4816	(84194)
Gamma	16	1	44	—	1685	(83715)
		0.01	49	7915	4801	(84445)
Delta	16	1	40	—	1116	(57631)
		0.01	46	9492	4865	(84421)
Epsilon	32	1	730	—	17436	(58488)
		0.01	293	—	36319	(56984)
Zeta	32	1	544	—	14368	(22814)
		0.01	297	—	37283	(68059)
FD	32	1	3199	—	—	(39227)
		0.01	2152	—	—	(52408)
OCR	32	1	1361	—	—	(58307)
		0.01	1330	—	—	(36523)
DNA	48	1	2668	—	—	—
		0.01	6557	—	—	14821

Dataset	C	OOPS n	OOPS # cores	OOPS Time	LibLinear n	LibLinear Time	LaRank n	LaRank Time
Alpha	1	500,000	16	39	500,000	147	500,000	3354
	0.01			50		112		2474
Beta	1	500,000	16	120	500,000	135	500,000	6372
	0.01			48		112		1880
Gamma	1	500,000	16	44	500,000	(8845)	500,000	—
	0.01			49		348		20318
Delta	1	500,000	16	40	500,000	(13266)	500,000	—
	0.01			46		429		—
Epsilon	1	500,000	32	730	250,000	316	500,000	5599
	0.01			293		265		2410
Zeta	1	500,000	32	544	250,000	278	500,000	—
	0.01			297		248		—
FD	1	2,560,000	32	3199	500,000	231	500,000	1537
	0.01			2152		193		332
OCR	1	3,500,000	32	1361	250,000	181	500,000	5695
	0.01			1330		121		4266
DNA	1	6,000,000	48	2668	600,000	144	600,000	300
	0.01			6557		30		407

Accuracy measured using area under precision recall curve.

Evaluation results taken from PASCAL Challenge website.

Dataset	OOPS	LibLinear	LaRank
Alpha	**0.1345**	0.1601	0.1606
Beta	**0.4988**	**0.4988**	0.5001
Gamma	**0.1174**	0.1185	0.1187
Delta	**0.1344**	0.1346	0.1355
Epsilon	**0.0341**	0.4935	0.4913
Zeta	**0.0115**	0.4931	0.4875
FD	**0.2274**	0.2654	0.3081
OCR	**0.1595**	0.1660	0.1681

Nonlinear and/or Indefinite Kernels:

A **kernel** is a function K, such that for all $x_i, x_j \in X$

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle,$$

where ϕ is a mapping from X to an (inner product) feature space F. We use $\langle ., . \rangle$ to denote a scalar product.

Linear Kernel $K(x_i, x_j) = x_i^T x_j$.

Polynomial Kernel $K(x_i, x_j) = (x_i^T x_j + 1)^d$.

Gaussian Kernel $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$.

Nonlinear and/or Indefinite Kernels:

Kernels are in general dense matrices making large-scale SVM training computationally demanding (or impossible).

Challenge: How to approximate kernels?

Find $\tilde{Q}$ with "desirable" properties such that

$$distance(Q, \tilde{Q})$$

is minimized. The distance may be measured with a matrix norm (say, Frobenius) or a Bregman divergence.

Dhillon and Tropp, Matrix nearness problems with Bregman divergences, *SIAM J. on Matrix An. & Appl.*, 29(2007) 1120-1146.

Lanckriet, Cristianini, Bartlett, Ghaoui and Jordan, *Learning the kernel matrix with semidefinite programming*, Journal of Machine Learning Research 5 (2004), 27-72.

IPM perspective: nonlinear/indefinite Kernels:

Approximate kernel matrix $Q, Q_{ij} = K(x_i, x_j)$ using low rank outer product

$$Q \approx L\Lambda L^T + D,$$

where $L \in \mathcal{R}^{n\times k},\ k \ll n.$

Exploit separability within IPMs

Augmented system becomes:

$$H = \begin{bmatrix} & & L \\ & & \\ L^T & & \end{bmatrix}$$

Conclusions:

Interior Point Methods

$\rightarrow$ **are well-suited to large scale optimization**

Support Vector Machine training

$\rightarrow$ **requires a solution of very large optimization problem**

IPMs provide an attractive approach to solve SVM training problems

Thank you for your attention!

Woodsend and Gondzio,
Exploiting separability in large-scale linear SVM training,
Tech Rep MS-07-002, Edinburgh 2007.
`http://www.maths.ed.ac.uk/~gondzio/reports/wgSVM.html`

Woodsend and Gondzio,
Hybrid MPI/OpenMP parallel linear SVM training,
Tech Rep ERGO-09-001, Edinburgh, 2009.
`http://www.maths.ed.ac.uk/~gondzio/reports/wgHybridSVM.html`