Machine Learning with GPUs:

Fast Support Vector Machines without the Coding Headaches

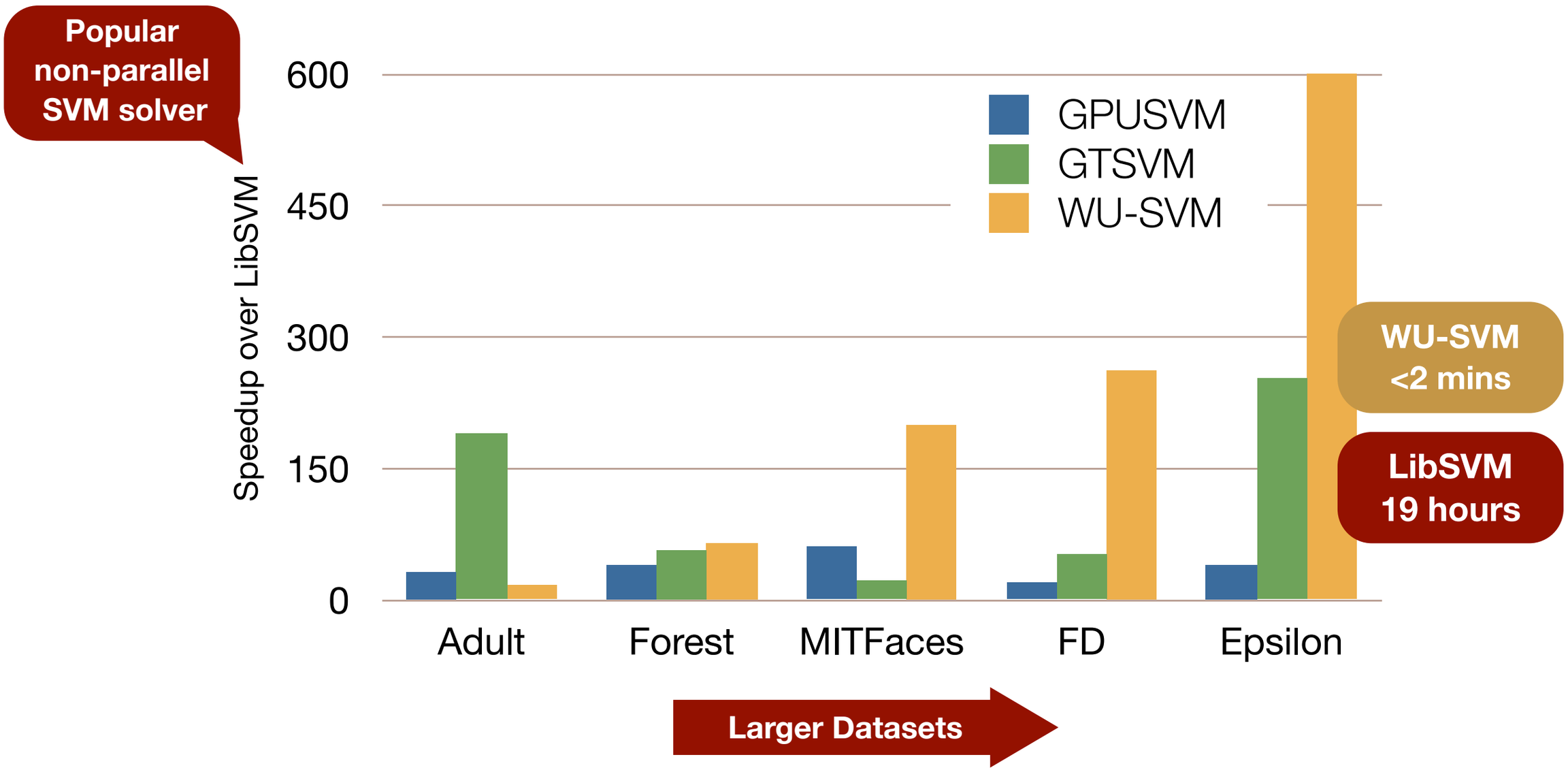
Stephen Tyree
PhD Student
Washington University in St. Louis

NVIDIA GPU Technology Conference
March 25, 2014

machinelearning.wustl.edu

Washington University in St. Louis

Starting with the last slide...



Takeaways

A bit of machine learning...

Takeaways

A bit of machine learning...

An example of choosing a “high-efficiency” algorithm (plays well with great code like CUBLAS)...

Takeaways

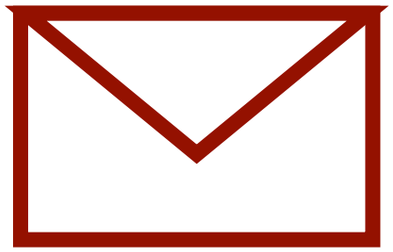
A bit of machine learning...

An example of choosing a “high-efficiency” algorithm (plays well with great code like CUBLAS)...

Some really fast SVM training code...

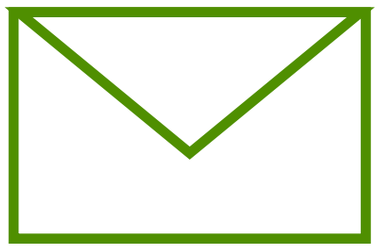
Machine Learned Classification

Learn to distinguish two types of things...



“Spam” emails

vs



“Ham” emails

Handwritten 3s

vs

Handwritten 4s

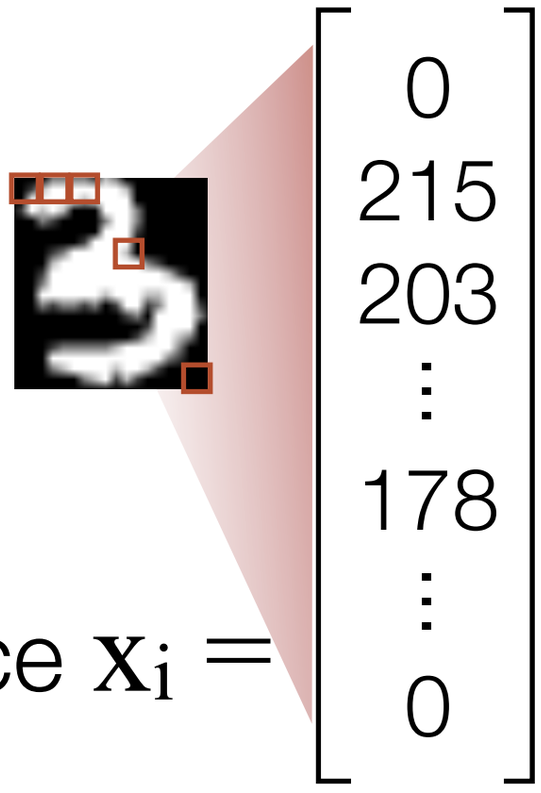
My face

vs

Other faces

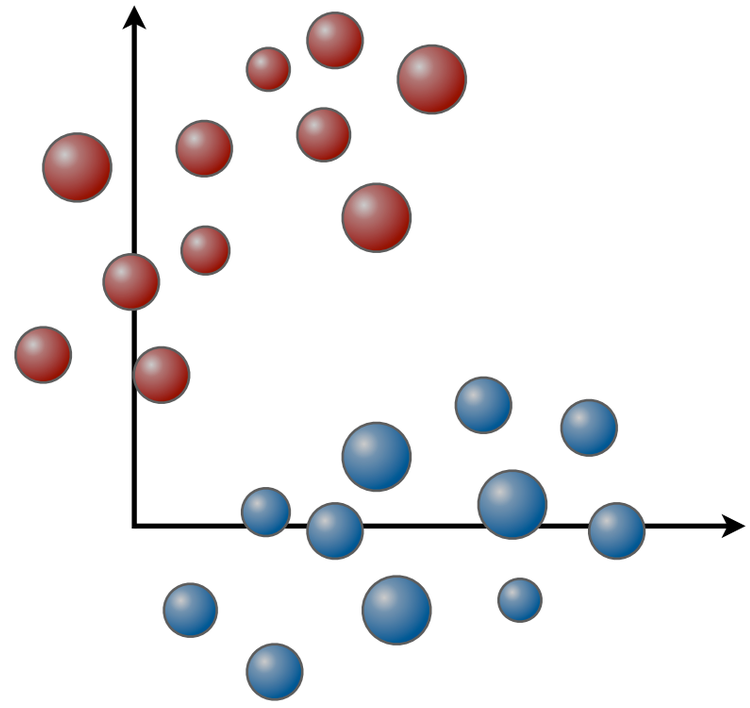
Machine Learned Classification

Vector feature representation



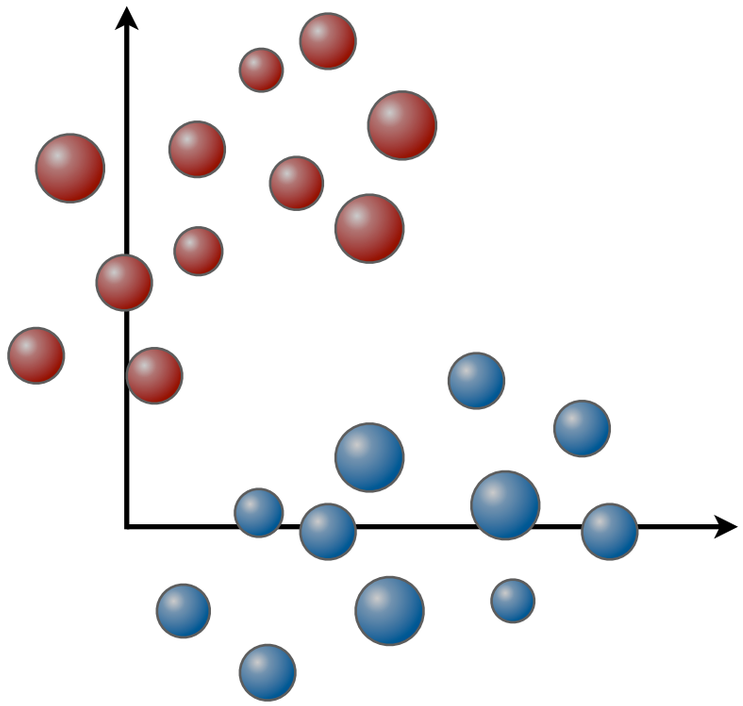
$$\text{Instance } \mathbf{x}_i = \begin{bmatrix} 0 \\ 215 \\ 203 \\ \vdots \\ 178 \\ \vdots \\ 0 \end{bmatrix}$$

$$\text{Label } \mathbf{y}_i = +1$$



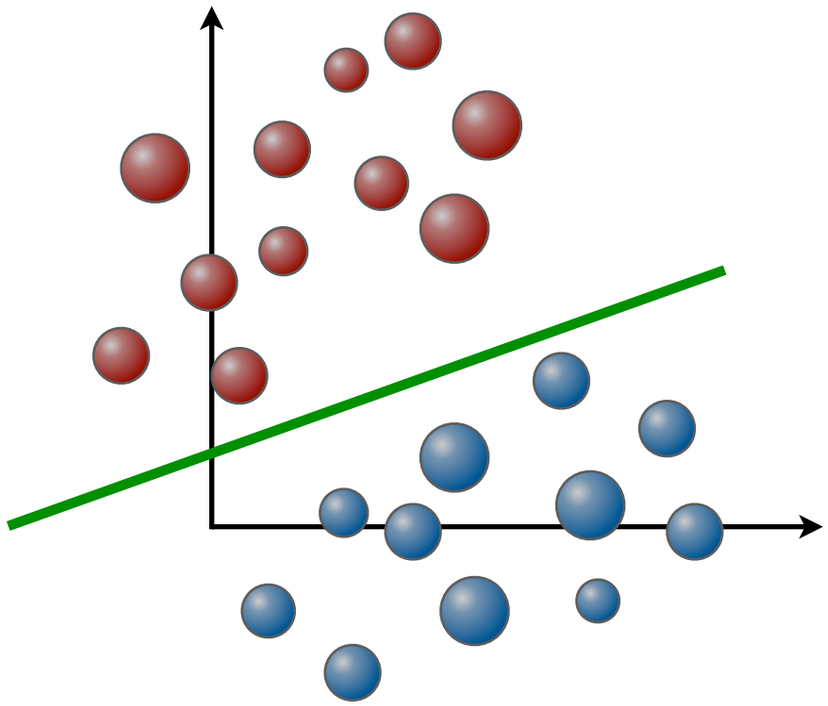
Machine Learned Classification

Training data



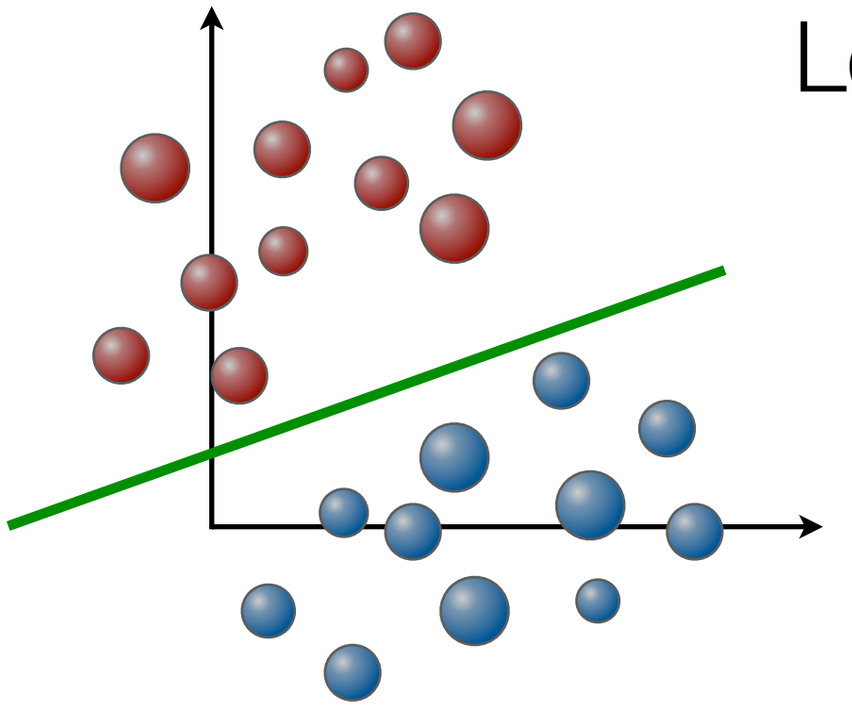
Machine Learned Classification

Training data



Machine Learned Classification

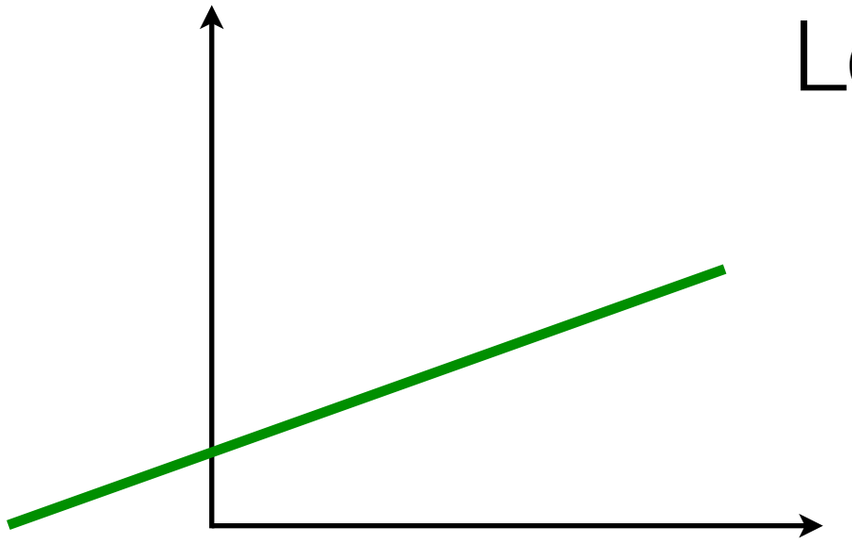
Training data



Learn model $(\mathbf{w},\mathbf{b})$

$$\mathbf{w}^\top \mathbf{x}+\mathbf{b}=0$$

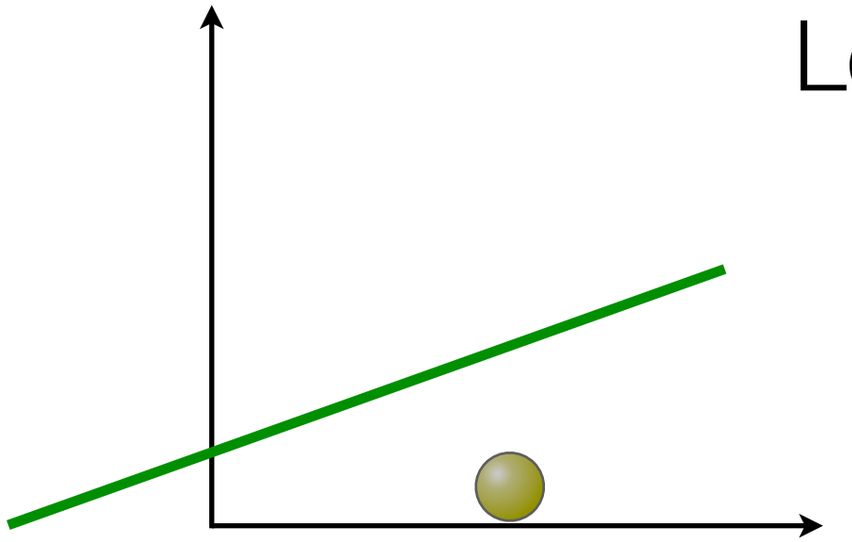
Machine Learned Classification



Learn model $(\mathbf{w},\mathbf{b})$

$$\mathbf{w}^\top \mathbf{x}+\mathbf{b} = 0$$

Machine Learned Classification

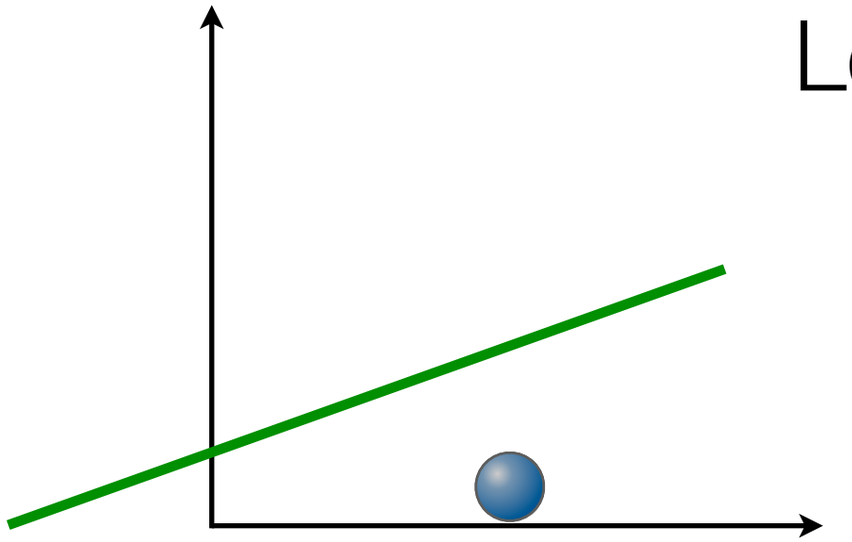


Learn model $(\mathbf{w},\mathbf{b})$

$$w^\top x+b = 0$$

$$\text{sign}(w^\top x+b) > 0?$$

Machine Learned Classification



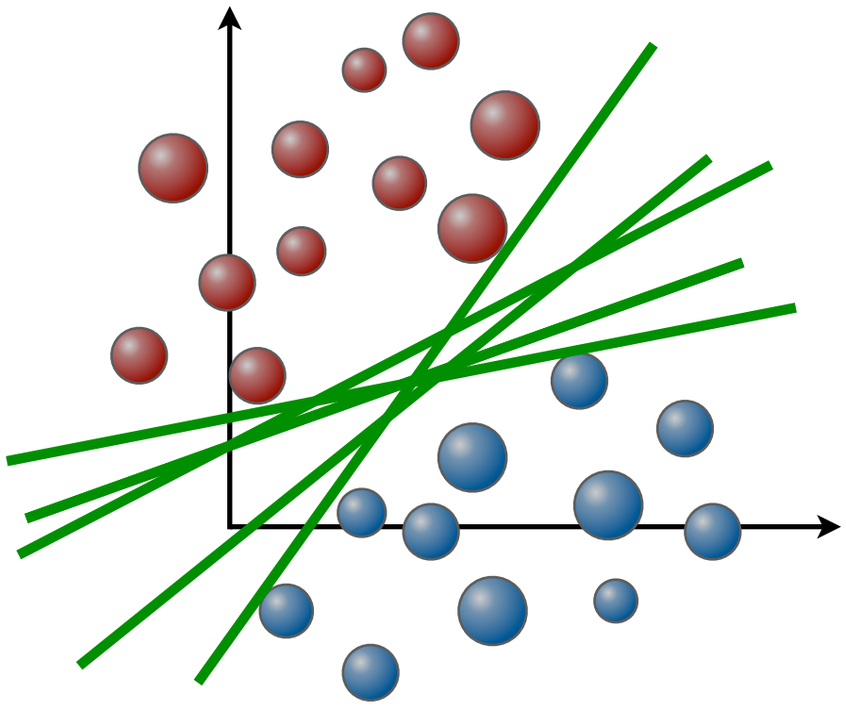
Learn model $(\mathbf{w},\mathbf{b})$

$$w^\top x+b = 0$$

$$\text{sign}(w^\top x+b) > 0?$$

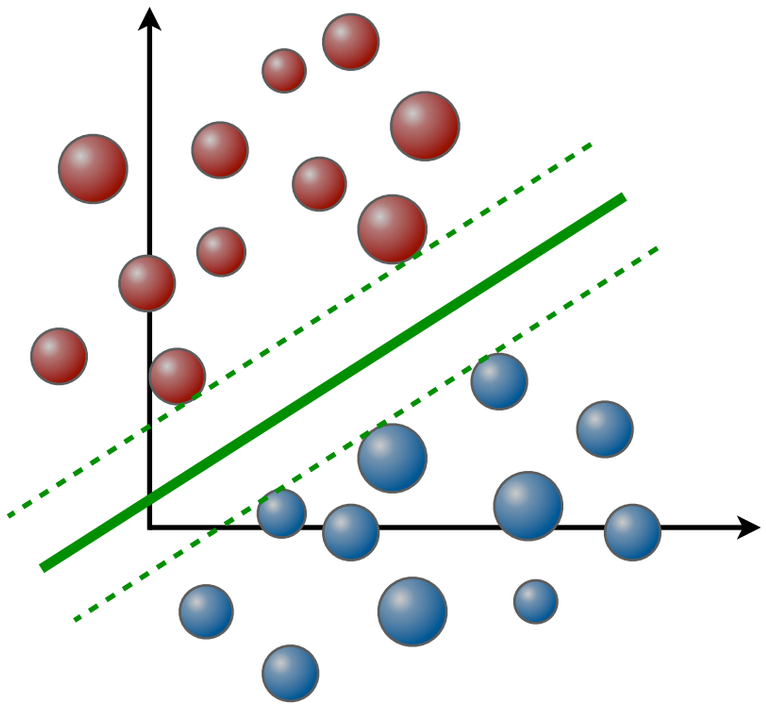
Machine Learned Classification

Which model?



Support Vector Machines [Cortez & Vapnik, 1995]

"Maximum margin"



"Margin" term

Error penalty

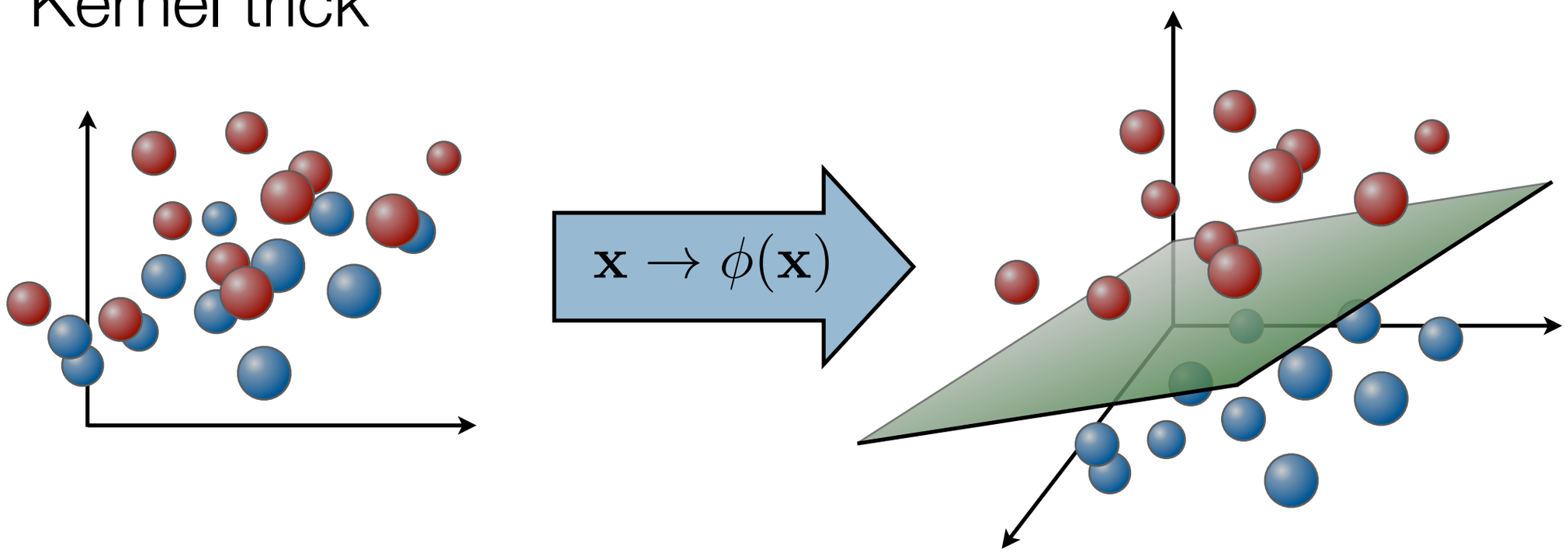
$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b))$$

True Label

Prediction

Kernel Support Vector Machines

"Kernel trick"



Non-linear feature projection + linear model =
non-linear decision boundary

Kernel Support Vector Machines

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b))$$

Kernel Support Vector Machines

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b))$$

high/infinite dimensionality

Kernel Support Vector Machines

high/infinite dimensionality

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b))$$

Dual

$$\max_{C \geq \alpha_i \geq 0} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \, \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) + \sum_{i=1}^{n} \alpha_i$$

$$s.t. \sum_{i=1}^{n} \alpha_i y_i = 0$$

Kernel Support Vector Machines

high/infinite dimensionality

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b))$$

Dual

$$\max_{C \geq \alpha_i \geq 0} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) + \sum_{i=1}^{n} \alpha_i$$

$$s.t. \sum_{i=1}^{n} \alpha_i y_i = 0$$

Kernel Support Vector Machines

$$\max_{C \geq \alpha_i \geq 0} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0$$

Kernel Support Vector Machines

$$\max_{C \geq \alpha_i \geq 0} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

Kernel Support Vector Machines

$$\max_{C \geq \alpha_i \geq 0} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2}$$

e.g. RBF kernel

Kernel Support Vector Machines

$$\max_{C \geq \alpha_i \geq 0} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0$$

Kernel function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2}$$

e.g. RBF kernel

$$\max_{C \geq \alpha_i \geq 0} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0$$

Sequential minimal optimization [Platt, 1998]

$$\max_{C \geq \alpha_i \geq 0} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0$$

Quadratic program,
$O(n^2)$ time & space

Sequential minimal optimization [Platt, 1998]

$$\max_{C \geq \alpha_i \geq 0} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0$$

Quadratic program, $O(n^2)$ time & space

Heuristically picks two variables to optimize in closed form

Sequential minimal optimization [Platt, 1998]

$$\max_{C \geq \alpha_i \geq 0} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0$$

Quadratic program, $O(n^2)$ time & space

Heuristically picks two variables to optimize in closed form

Only two kernel matrix rows needed per iteration

Sequential minimal optimization [Platt, 1998]

$$\max_{C \geq \alpha_i \geq 0} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0$$

Quadratic program, $O(n^2)$ time

Heuristically picks two variables to optimize in closed form

Only two kernel matrix rows needed per iteration

Sequential minimal optimization [Platt, 1998]

$$\max_{C \geq \alpha_i \geq 0} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0$$

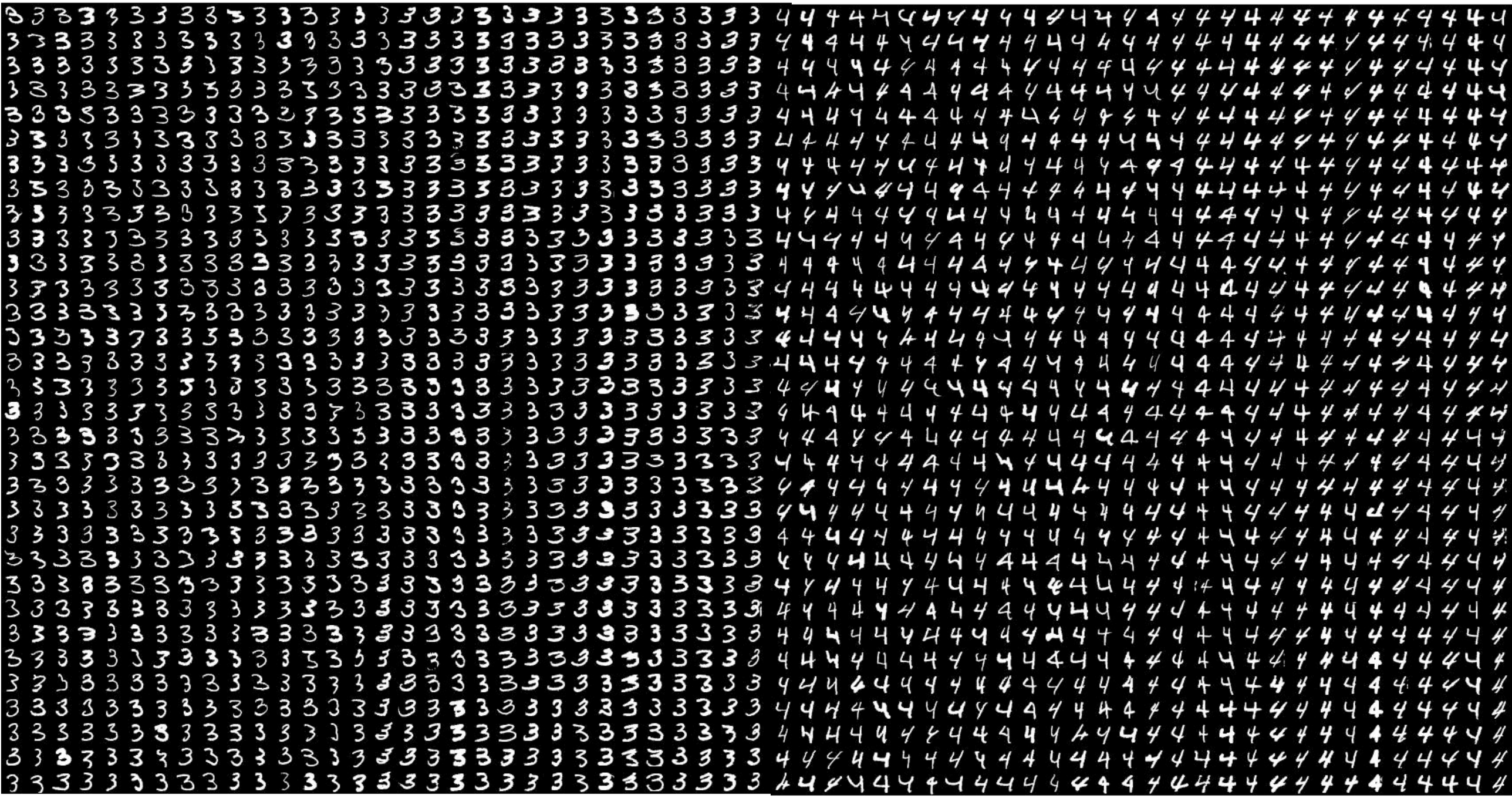
Quadratic program, $O(n^2)$ time

Heuristically picks two variables to optimize in closed form

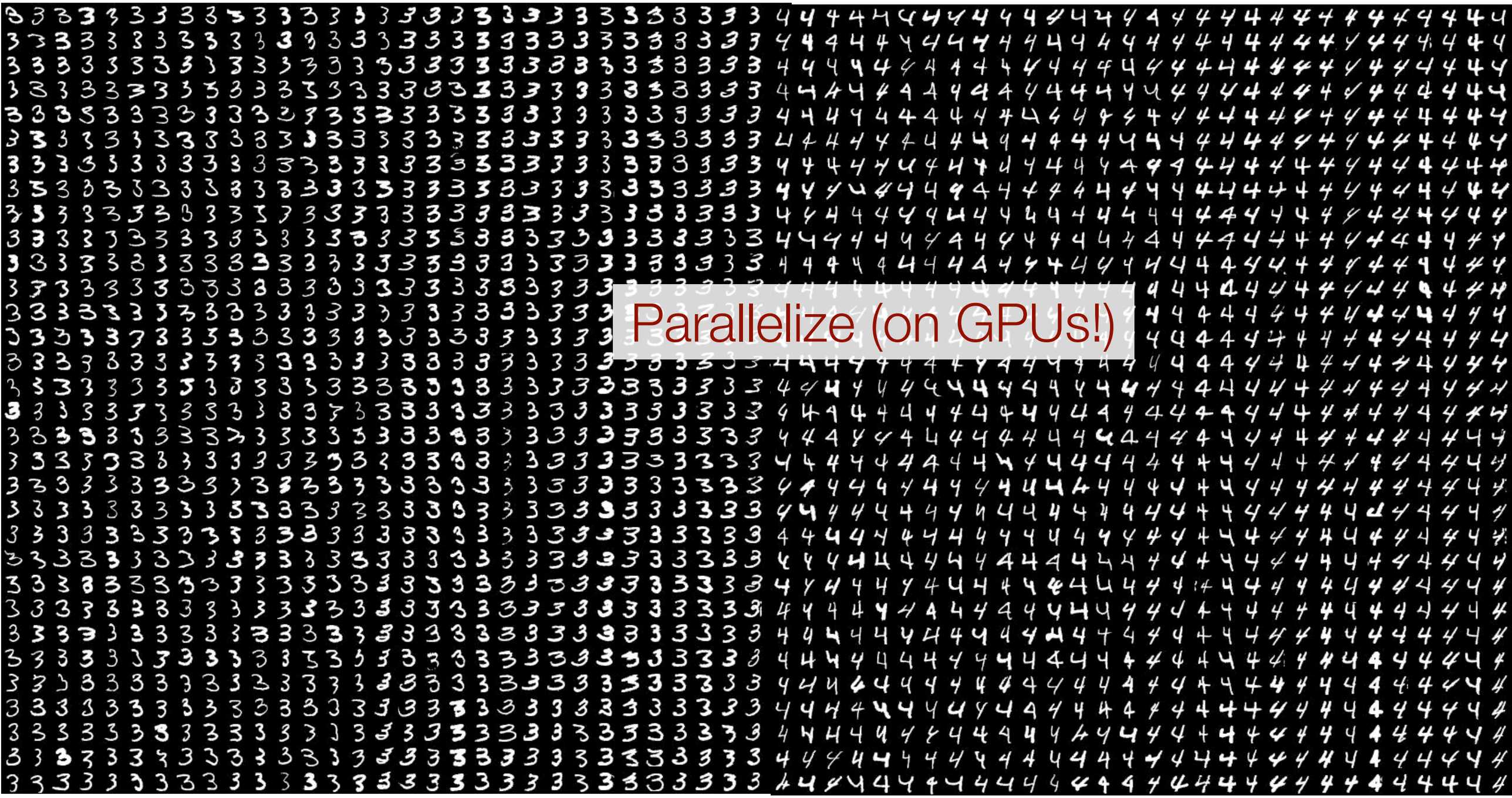
Only two kernel matrix rows needed per iteration

Many lightweight iterations

What about large training sets?

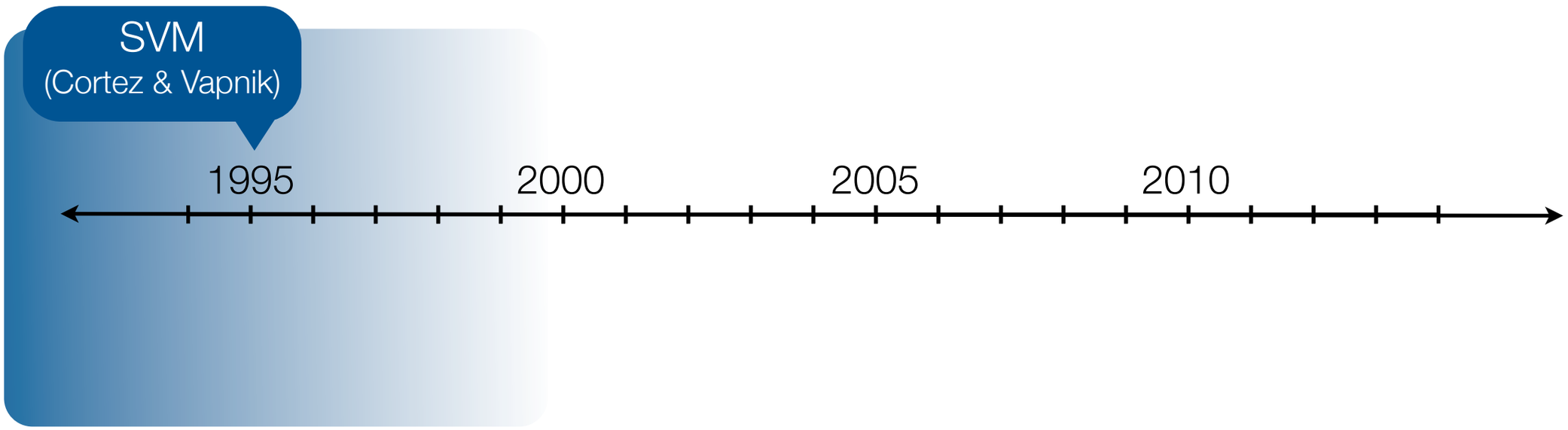


What about large training sets?



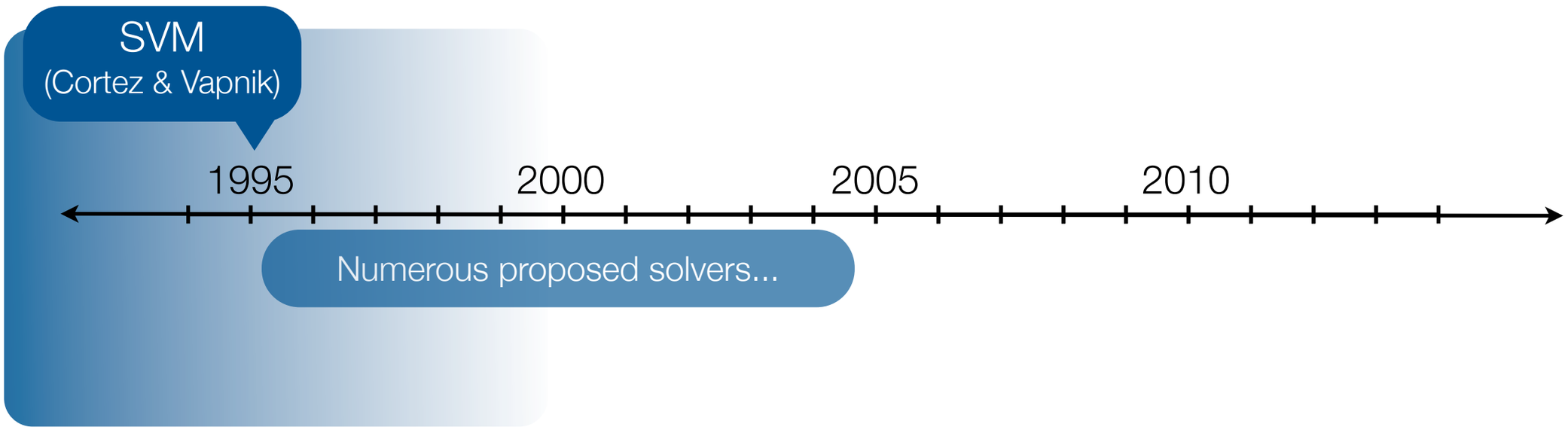
Can we do better?

- Are dual decomposition methods the best candidates for parallelization?
  - well-known and understood solvers
  - but, many lightweight iterations
  - complex hand-coded solutions



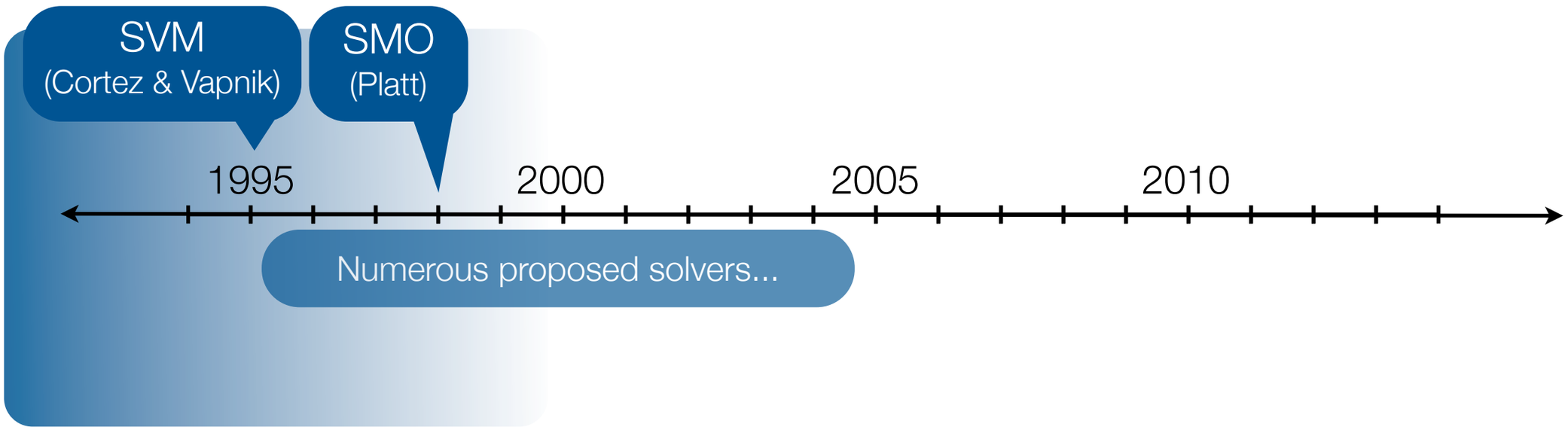
Can we do better?

- Are dual decomposition methods the best candidates for parallelization?
  - well-known and understood solvers
  - but, many lightweight iterations
  - complex hand-coded solutions



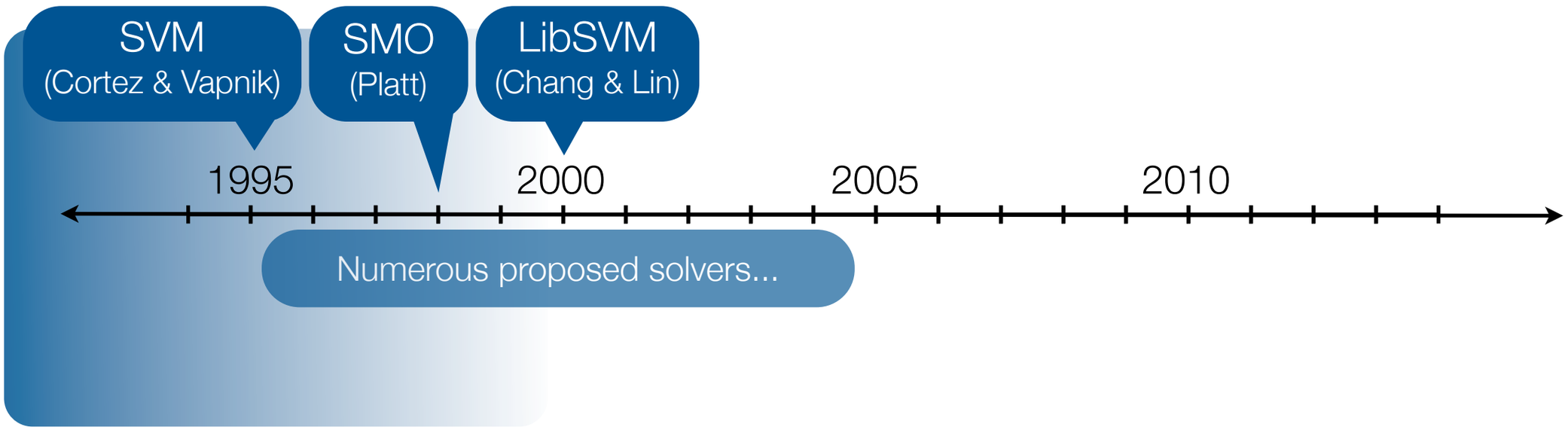
Can we do better?

- Are dual decomposition methods the best candidates for parallelization?
  - well-known and understood solvers
  - but, many lightweight iterations
  - complex hand-coded solutions



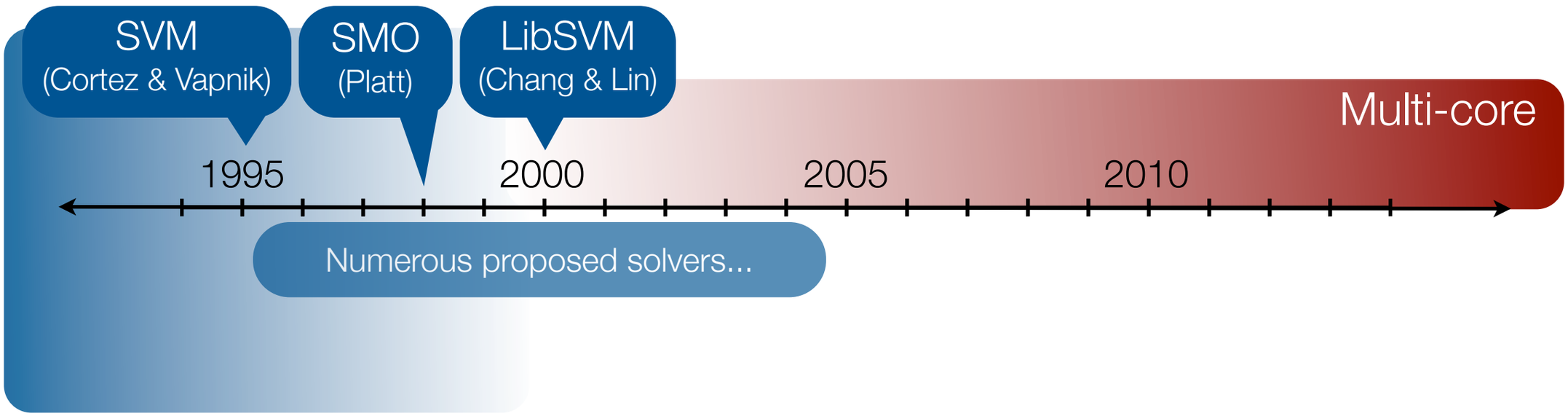
Can we do better?

- Are dual decomposition methods the best candidates for parallelization?
  - well-known and understood solvers
  - but, many lightweight iterations
  - complex hand-coded solutions



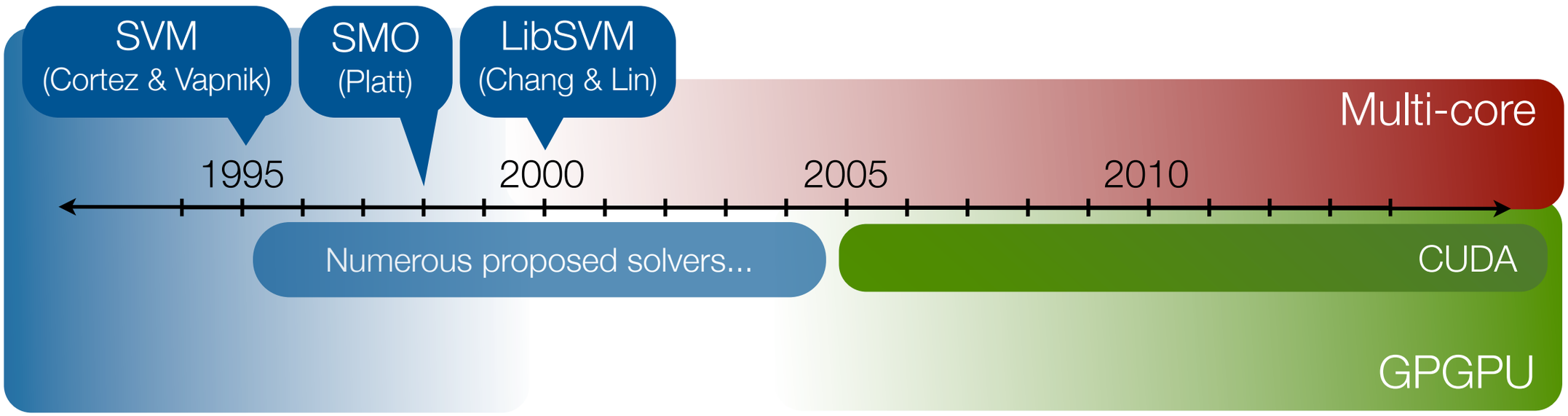
Can we do better?

- Are dual decomposition methods the best candidates for parallelization?
  - well-known and understood solvers
  - but, many lightweight iterations
  - complex hand-coded solutions



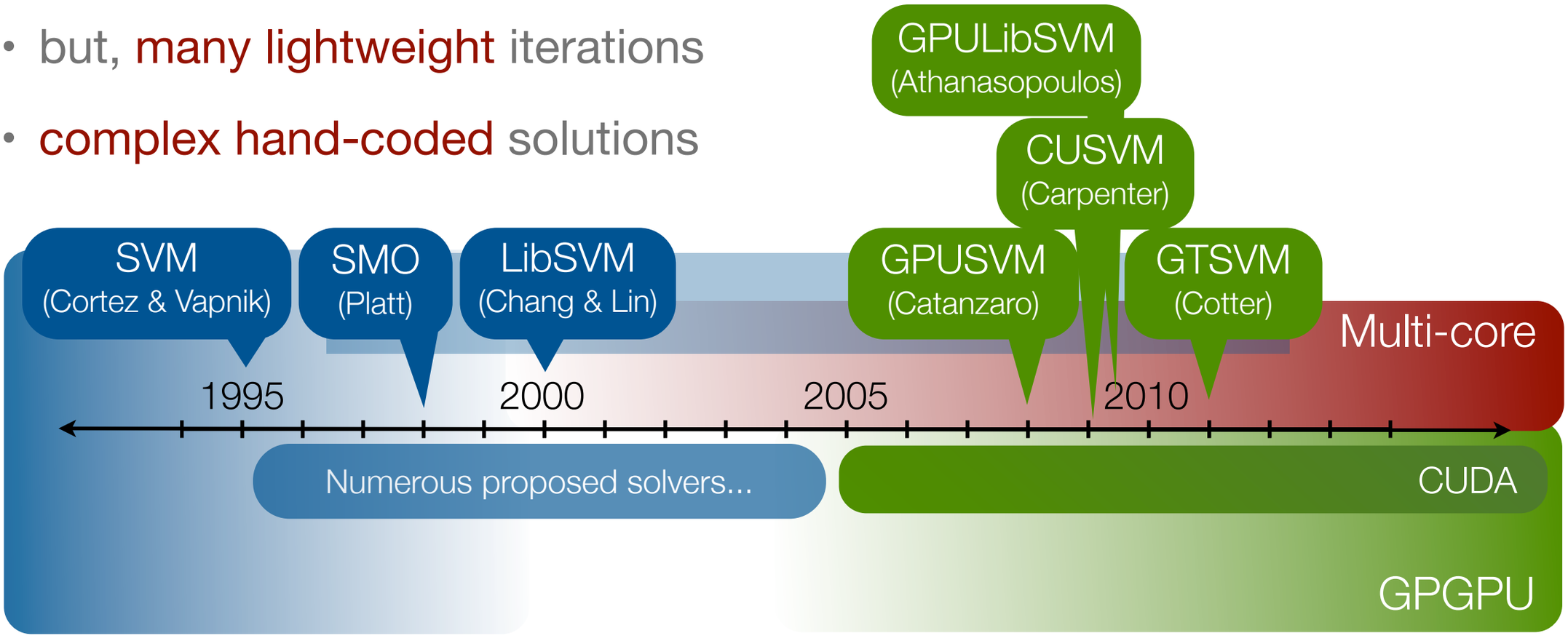
Can we do better?

- Are dual decomposition methods the best candidates for parallelization?
  - well-known and understood solvers
  - but, many lightweight iterations
  - complex hand-coded solutions



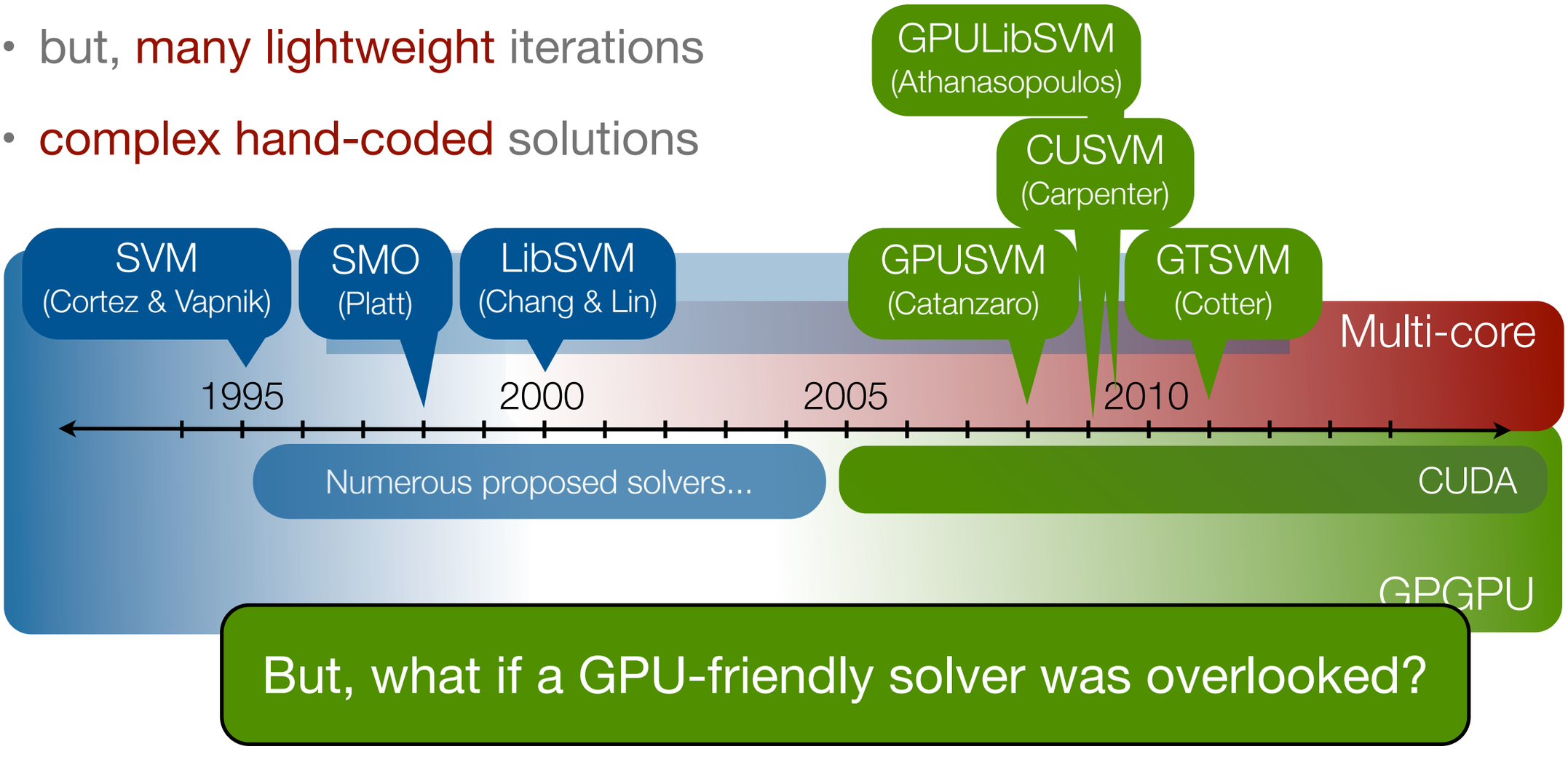
Can we do better?

- Are dual decomposition methods the best candidates for parallelization?
  - well-known and understood solvers
  - but, many lightweight iterations
  - complex hand-coded solutions



Can we do better?

- Are dual decomposition methods the best candidates for parallelization?
  - well-known and understood solvers
  - but, many lightweight iterations
  - complex hand-coded solutions



Can we do better?

- Literature search

  - ☐ few, dense iterations

  - ☐ easily parallelized core computations (e.g. linear algebra)

  - ☐ approximation (since traditional SVMs scale number of parameters with the number of training instances)

Can we do better?

- Literature search

  - ☐ few, dense iterations

  - ☐ easily parallelized core computations (e.g. linear algebra)

  - ☐ approximation (since traditional SVMs scale number of parameters with the number of training instances)

Sparse Primal SVM
[Keerthi,2006]

Solves approximate SVM in primal form using the representer theorem [Chapelle, 2000]

Can we do better?

- Literature search

  - [x] few, dense iterations -- amenable to Newton steps (fast convergence!)
  - [ ] easily parallelized core computations (e.g. linear algebra)
  - [ ] approximation (since traditional SVMs scale number of parameters with the number of training instances)

Sparse Primal SVM
[Keerthi,2006]

Solves approximate SVM in primal form using the representer theorem [Chapelle, 2000]

Can we do better?

- Literature search

  ☑ few, dense iterations -- amenable to Newton steps (fast convergence!)

  ☑ easily parallelized core computations (e.g. linear algebra)
  -- each step is computed with dense linear algebra

  ☐ approximation (since traditional SVMs scale number of parameters with the number of training instances)

Sparse Primal SVM
[Keerthi,2006]

Solves approximate SVM in primal form using the representer theorem
[Chapelle, 2000]

Can we do better?

- Literature search

  ☑ few, dense iterations -- amenable to Newton steps (fast convergence!)

  ☑ easily parallelized core computations (e.g. linear algebra)
  -- each step is computed with dense linear algebra

  ☑ approximation (since traditional SVMs scale number of parameters with the number of training instances)
  -- control approximation by number of kernel basis vectors

Sparse Primal SVM
[Keerthi,2006]

Solves approximate SVM in primal form using the representer theorem
[Chapelle, 2000]

Sparse Primal SVM [Keerthi, 2006]

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b))$$

Original SVM Objective

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

Sparse Primal SVM [Keerthi, 2006]

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b))$$

Original SVM Objective

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

Change of variable

$$\mathbf{w} = \sum_{j \in \mathcal{J}} \beta_j \phi(\mathbf{x}_j)$$

Small basis set ("support vectors")

Sparse Primal SVM [Keerthi, 2006]

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b))$$

Original SVM Objective

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

Change of variable

$$\mathbf{w} = \sum_{j \in \mathcal{J}} \beta_j \phi(\mathbf{x}_j)$$

Small basis set ("support vectors")

$$\min_{\boldsymbol{\beta},b} \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}_{\mathcal{J}\mathcal{J}}\boldsymbol{\beta} + \frac{C}{2}\sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{\beta}^\top \mathbf{k}_{\mathcal{J}i} + b))^2$$

SP-SVM Objective

Sparse Primal SVM [Keerthi, 2006]

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b))$$

Original SVM Objective

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

Change of variable

$$\mathbf{w} = \sum_{j \in \mathcal{J}} \beta_j \phi(\mathbf{x}_j)$$

Small basis set ("support vectors")

$$\min_{\boldsymbol{\beta},b} \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}_{\mathcal{J}\mathcal{J}}\boldsymbol{\beta} + \frac{C}{2}\sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{\beta}^\top \mathbf{k}_{\mathcal{J}i} + b))^2$$

SP-SVM Objective

Sparse Primal SVM [Keerthi, 2006]

SP-SVM Objective

$$\min_{\boldsymbol{\beta},b} \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}_{\mathcal{J}\mathcal{J}}\boldsymbol{\beta} + \frac{C}{2}\sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{\beta}^\top \mathbf{k}_{\mathcal{J}i} + b))^2$$

Sparse Primal SVM [Keerthi, 2006]

SP-SVM Objective

$$\min_{\boldsymbol{\beta},b} \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}_{\mathcal{J}\mathcal{J}}\boldsymbol{\beta} + \frac{C}{2}\sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{\beta}^\top \mathbf{k}_{\mathcal{J}i} + b))^2$$

1xB

BxN

$$\mathbf{g} = \mathbf{K}_{\mathcal{J}\mathcal{J}}\boldsymbol{\beta} - C\mathbf{K}_{\mathcal{J}\mathcal{I}}(\mathbf{y}_{\mathcal{I}} - \boldsymbol{\beta}^\top \mathbf{K}_{\mathcal{J}\mathcal{I}})$$

Gradient

$$\mathbf{H} = \mathbf{K}_{\mathcal{J}\mathcal{J}} + C\mathbf{K}_{\mathcal{J}\mathcal{I}}\mathbf{K}_{\mathcal{J}\mathcal{I}}^\top$$

Hessian

Matrix/Vector Multiply

Matrix/Matrix Multiply

Linear System Solver

Sparse Primal SVM [Keerthi, 2006]

SP-SVM Objective

$$\min_{\boldsymbol{\beta},b} \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}_{\mathcal{J}\mathcal{J}}\boldsymbol{\beta} + \frac{C}{2}\sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{\beta}^\top \mathbf{k}_{\mathcal{J}i} + b))^2$$

1xB

BxN

$$\mathbf{g} = \mathbf{K}_{\mathcal{J}\mathcal{J}}\boldsymbol{\beta} - C\mathbf{K}_{\mathcal{J}\mathcal{I}}(\mathbf{y}_{\mathcal{I}} - \boldsymbol{\beta}^\top \mathbf{K}_{\mathcal{J}\mathcal{I}})$$

Gradient

$$\mathbf{H} = \mathbf{K}_{\mathcal{J}\mathcal{J}} + C\mathbf{K}_{\mathcal{J}\mathcal{I}}\mathbf{K}_{\mathcal{J}\mathcal{I}}{}^\top$$

Hessian

BxB

1xB

$$\boldsymbol{\beta}' = \boldsymbol{\beta} - \mathbf{H}^{-1}\mathbf{g}$$

Newton Optimization

Matrix/Vector Multiply

Matrix/Matrix Multiply

Linear System Solver

Implementation

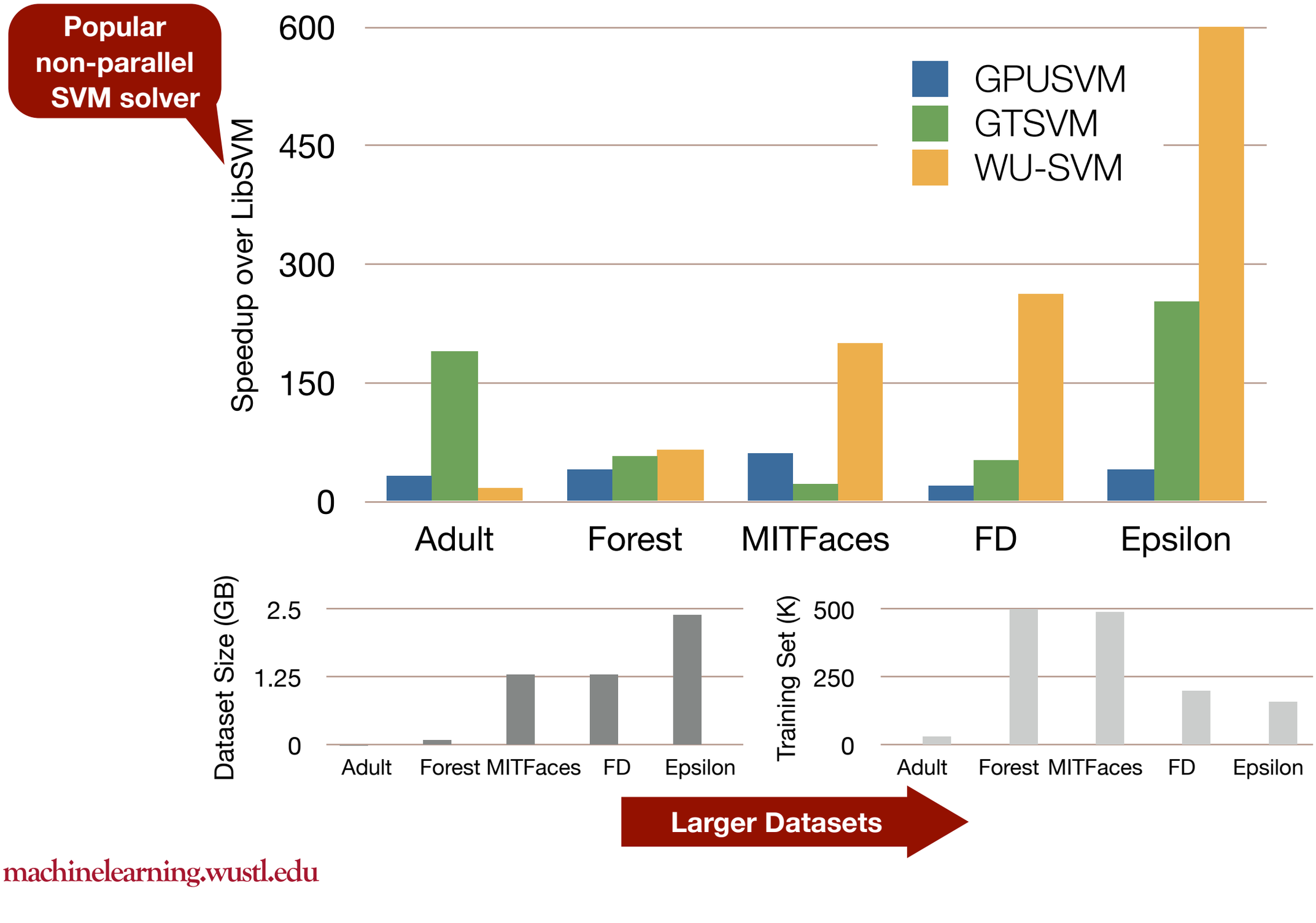
- CUBLAS!
  - No tedious hand-coding of kernels
  - Takes advantage of insider knowledge (CUBLAS by NVIDIA)
  - Automatically updates to latest graphics cards and BLAS library updates
  - (Almost) automatically transfers to other platforms (with BLAS implementations, i.e. multicore)

WU-SVM

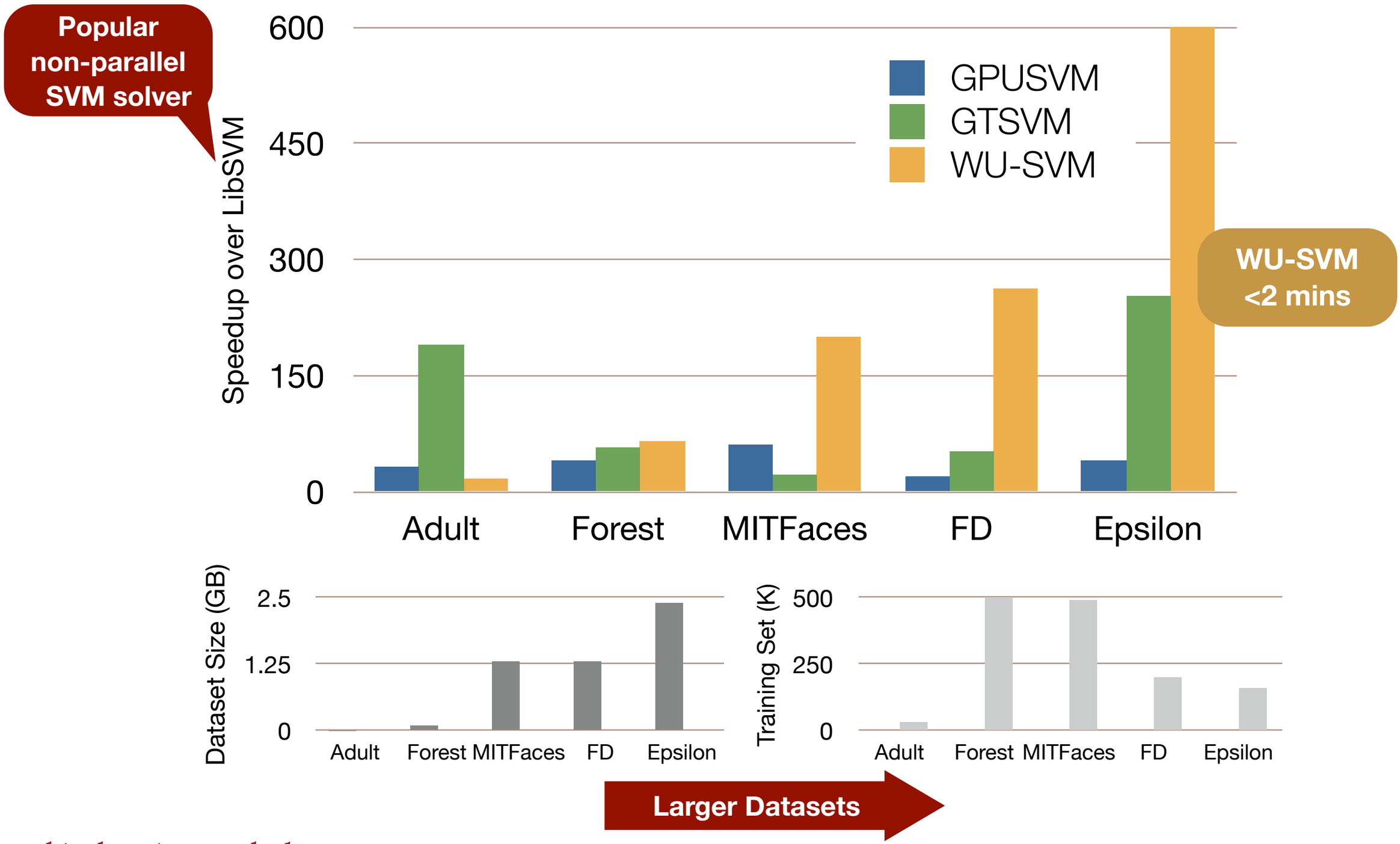
- SP-SVM in C++/(CU)BLAS with open source BSD license
- Compatible with multicore BLAS implementations
- Implements LibSVM interface (command line, dataset format, output files)
- Written by undergraduate coders: Gabriel Hope, Nicholas Kolkin, and Jack Hessel

http://www.tinyurl.com/wu-svm

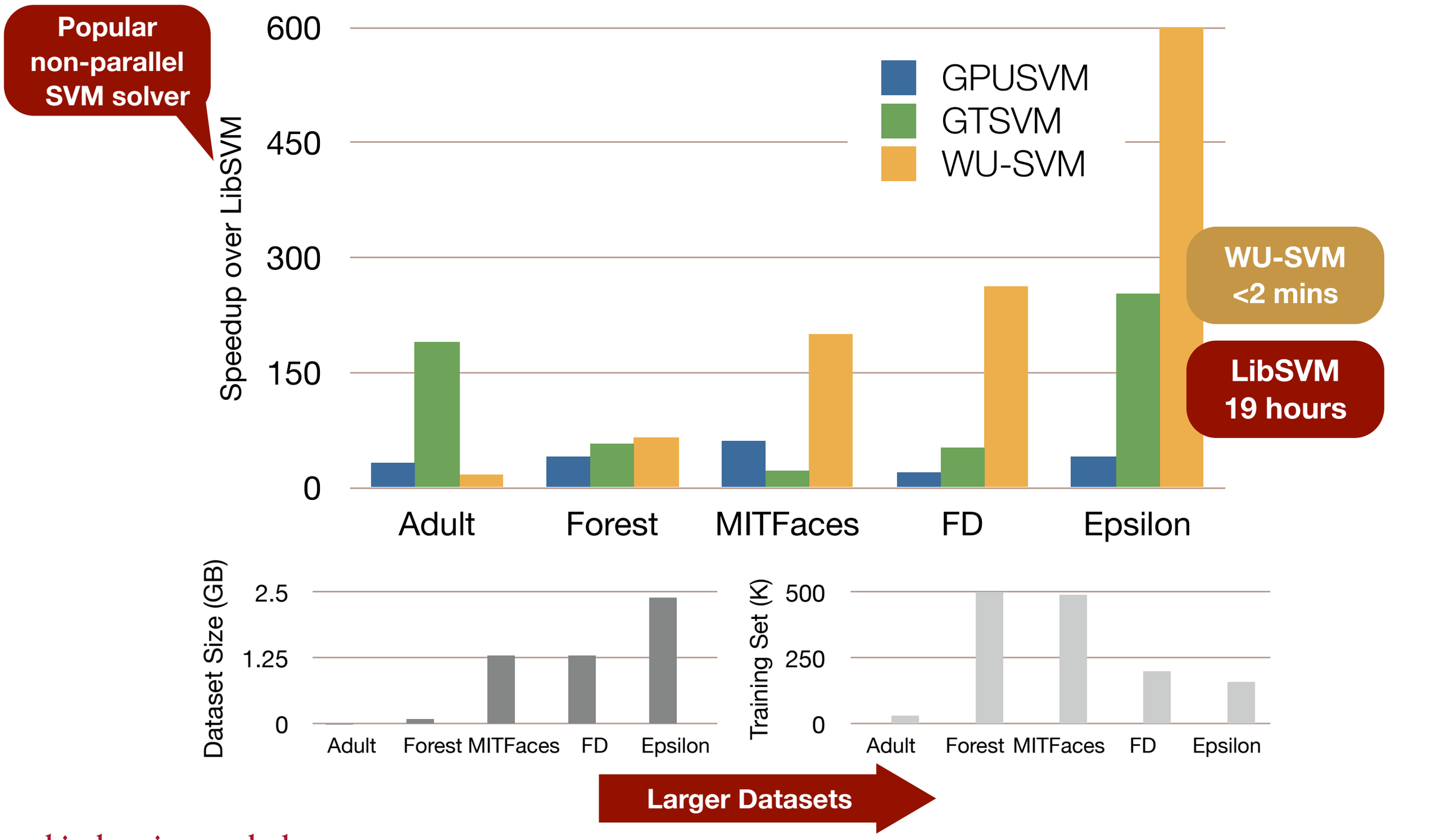
Experiments



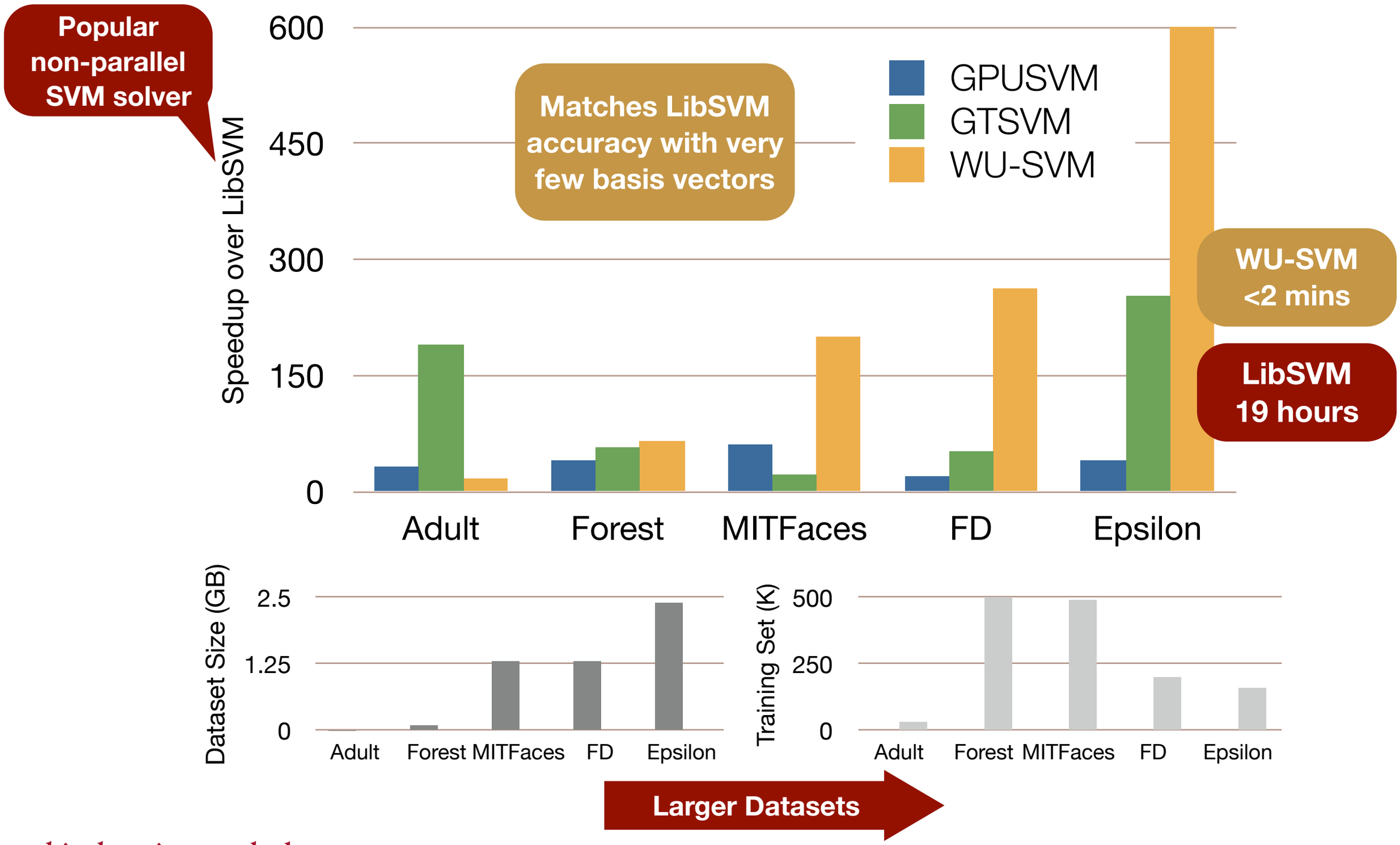
Experiments



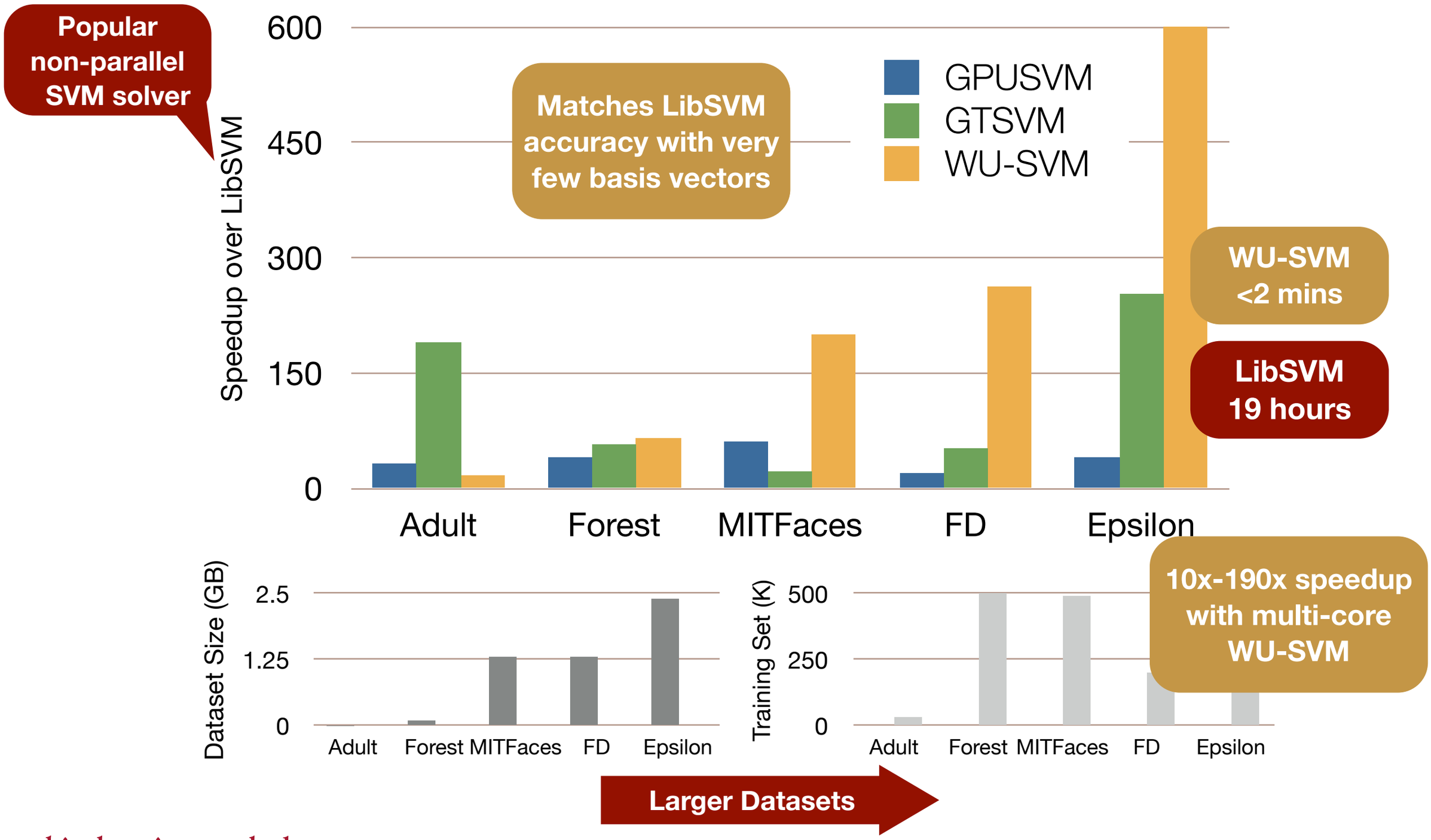
Experiments



Experiments



Experiments



Conclusions

- Why WU-SVM?

Conclusions

- Why WU-SVM?
  - Highly parallel since it's the right type of operation (largely dense linear algebra)

Conclusions

- Why WU-SVM?
  - Highly parallel since it's the right type of operation (largely dense linear algebra)
  - Very fast since well-designed/engineered code already exists

Conclusions

- Why WU-SVM?
  - Highly parallel since it's the right type of operation (largely dense linear algebra)
  - Very fast since well-designed/engineered code already exists
  - Easy to write/maintain/port to new platforms (since these libraries are standardized and constantly updated for new hardware)

Conclusions

- Why WU-SVM?
  - Highly parallel since it's the right type of operation (largely dense linear algebra)
  - Very fast since well-designed/engineered code already exists
  - Easy to write/maintain/port to new platforms (since these libraries are standardized and constantly updated for new hardware)
- All we did was put the right pieces together (Keerthi's SP-SVM + CUBLAS), but it makes you wonder what other methods could benefit from a similar approach...

http://www.tinyurl.com/wu-svm

Thank you! Questions?

Thanks to Gabriel Hope, Nicholas Kolkin, Jack Hessel, Jacob Gardner, John Tran (NVIDIA), Kunal Agrawal, and Kilian Weinberger

Stephen Tyree
swtyree@wustl.edu