Efficient multiplicative updates for SVM

Sergey Plis

Computer Science Department
University of New Mexico

NIMH grant number 1R01MH076282-01

joint work with Vamsi Potluru, Morten Mørup,
Vince Calhoun, and Terran Lane

Outline

Outline

Maximum Margin Classifiers

- Given two classes A and B Data is a set of labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$
- Bayesian Decision Boundary: *Distributions are known*

  $$g(\boldsymbol{x}) = P(A|\boldsymbol{x}) - P(B|\boldsymbol{x}) = 0$$

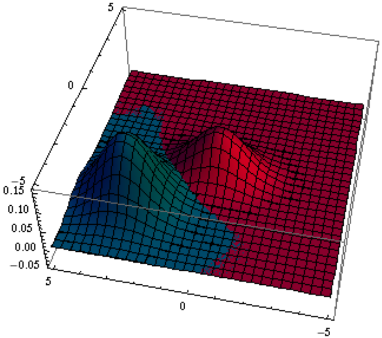
- Maximum Margin Hyperplane: *Only data is given*

Maximum Margin Classifiers

- Given two classes A and B Data is a set of labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$
- Bayesian Decision Boundary: *Distributions are known*

$$g(\boldsymbol{x}) = P(A|\boldsymbol{x}) - P(B|\boldsymbol{x}) = 0$$

- Maximum Margin Hyperplane: *Only data is given*

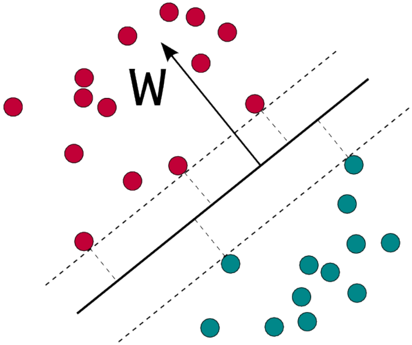


Maximum Margin Classifiers

- Given two classes A and B Data is a set of labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$
- Bayesian Decision Boundary: *Distributions are known*

$$g(\boldsymbol{x}) = P(A|\boldsymbol{x}) - P(B|\boldsymbol{x}) = 0$$

- Maximum Margin Hyperplane: *Only data is given*



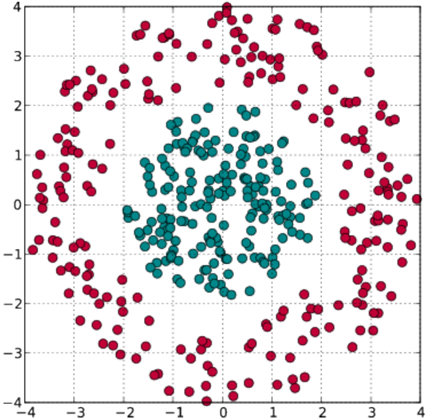
Mapping to Higher Dimensions

- A two class problem: not separable by a line
- Map each point into a higher dimensional space:

$$\eta_i = \Phi(\boldsymbol{x}_i)$$

- Choose Φ so the data becomes linearly separable

$$\Phi(\boldsymbol{x}) = (x_1^2, \sqrt(2)x_1x_2, x_2^2)$$



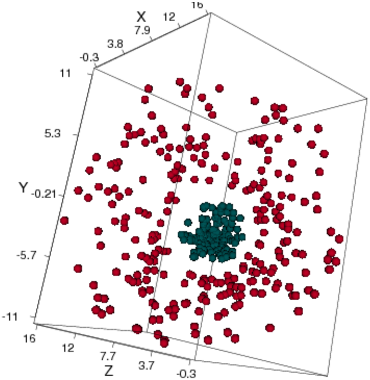
Mapping to Higher Dimensions

- A two class problem: not separable by a line
- Map each point into a higher dimensional space:

$$\boldsymbol{\eta}_i = \Phi(\boldsymbol{x}_i)$$

- Choose Φ so the data becomes linearly separable

$$\Phi(\boldsymbol{x}) = (x_1^2, \sqrt(2)x_1x_2, x_2^2)$$



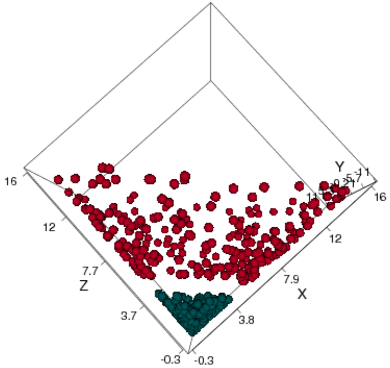
Mapping to Higher Dimensions

- A two class problem: not separable by a line
- Map each point into a higher dimensional space:

$$\eta_i = \Phi(\boldsymbol{x}_i)$$

- Choose Φ so the data becomes linearly separable

$$\Phi(\boldsymbol{x}) = (x_1^2, \sqrt(2)x_1 x_2, x_2^2)$$



Kernel trick

- Dimensions can be many (even ∞)!.Have to compute very expensive inner product?
- Avoid it by defining Hilbert spaces with kernels

$$\begin{aligned} \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle &= \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{y}) \\ (x_1^2, \sqrt{2}x_1x_2, x_2^2)(y_1^2, \sqrt{2}y_1y_2, y_2^2)^T &= (\boldsymbol{x} \cdot \boldsymbol{y})^2 \\ x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 &= (x_1y_1 + x_2y_2)^2 \end{aligned}$$

- Take a linear algorithm, replace inner products with kernels and get a nonlinear algorithm as a result!

Kernel trick

- Dimensions can be many (even ∞)!.Have to compute very expensive inner product?
- Avoid it by defining Hilbert spaces with kernels

$$
\begin{aligned}
\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle &= \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{y}) \\
(x_1^2, \sqrt{2}x_1x_2, x_2^2)(y_1^2, \sqrt{2}y_1y_2, y_2^2)^T &= (\boldsymbol{x} \cdot \boldsymbol{y})^2 \\
x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 &= (x_1y_1 + x_2y_2)^2
\end{aligned}
$$

- Take a linear algorithm, replace inner products with kernels and get a nonlinear algorithm as a result!

Kernel trick

- Dimensions can be many (even ∞)!.Have to compute very expensive inner product?
- Avoid it by defining Hilbert spaces with kernels

$$\begin{aligned}\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle &= \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{y}) \\ (x_1^2, \sqrt{2}x_1x_2, x_2^2)(y_1^2, \sqrt{2}y_1y_2, y_2^2)^T &= (\boldsymbol{x} \cdot \boldsymbol{y})^2 \\ x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 &= (x_1y_1 + x_2y_2)^2\end{aligned}$$

- Take a linear algorithm, replace inner products with kernels and get a nonlinear algorithm as a result!

Primal form

- Separating hyperplane

$$y = sign((\boldsymbol{w} \cdot \boldsymbol{x}) + b)$$

- Classification error

$$y_i((\boldsymbol{w} \cdot \Phi(\boldsymbol{x_i})) + b)$$

- Quadratic optimization problem

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{subject to } y_i((\boldsymbol{w} \cdot \Phi(\boldsymbol{x_i})) + b) \geq 1, i \in \{1..n\}$$

Dual form

- Introduce Lagrange multipliers:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i (y_i ((\boldsymbol{w} \cdot \Phi(\boldsymbol{x_i})) + b) - 1)$$

- The dual quadratic optimization problem for SVM Schölkopf and Smola [2001] is given by minimizing the following loss function:

$$S(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i$$

$$\text{subject to } \alpha_i \geq 0, i \in \{1..n\},$$

Dual form

- Introduce Lagrange multipliers:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i (y_i((\boldsymbol{w} \cdot \Phi(\boldsymbol{x_i})) + b) - 1)$$

- The dual quadratic optimization problem for SVM Schölkopf and Smola [2001] is given by minimizing the following loss function:

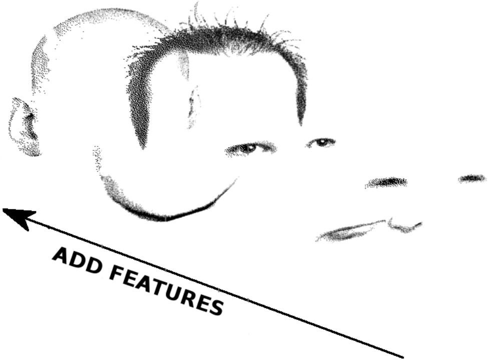
$$S(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i$$

$$\text{subject to } \alpha_i \geq 0, i \in \{1..n\},$$

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

Additive Features

- Features are non-negative and only add up
- Features are unknown: data comes as their combination



Additive Features

- Features are non-negative and only add up
- Features are unknown: data comes as their combination

Mathematical Formulation

- Given data $\boldsymbol{X}$ find its factorization:

$$\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}$$
$$\boldsymbol{X}_{ij} \geq 0 \; \boldsymbol{W}_{ij} \geq 0 \; \boldsymbol{H}_{ij} \geq 0$$

- Minimize the objective function:

$$E = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2$$

- Ignore other possible objectives

Mathematical Formulation

- Given data $\boldsymbol{X}$ find its factorization:

$$\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}$$
$$\boldsymbol{X}_{ij} \geq 0 \ \boldsymbol{W}_{ij} \geq 0 \ \boldsymbol{H}_{ij} \geq 0$$

- Minimize the objective function:

$$E = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2$$

- Ignore other possible objectives

Mathematical Formulation

- Given data $\boldsymbol{X}$ find its factorization:

$$\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}$$
$$\boldsymbol{X}_{ij} \geq 0 \; \boldsymbol{W}_{ij} \geq 0 \; \boldsymbol{H}_{ij} \geq 0$$

- Minimize the objective function:

$$E = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2$$

- Ignore other possible objectives

Gradient Descent

- Compute the derivative and find its zero

$$\frac{\partial E}{\partial \boldsymbol{W}} = \boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T - \boldsymbol{X}\boldsymbol{H}^T$$
$$\frac{\partial E}{\partial \boldsymbol{H}} = \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H} - \boldsymbol{W}^T\boldsymbol{X}$$

- Classical solution

$$\boldsymbol{H} = \boldsymbol{H} + \eta \odot (\boldsymbol{W}^T\boldsymbol{X} - \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H})$$

- Exponentiated gradient

$$\boldsymbol{H} = \boldsymbol{H} \odot e^{\eta \odot (\boldsymbol{W}^T\boldsymbol{X} - \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H})}$$

Gradient Descent

- Compute the derivative and find its zero

$$\frac{\partial E}{\partial \boldsymbol{W}} = \boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T - \boldsymbol{X}\boldsymbol{H}^T$$
$$\frac{\partial E}{\partial \boldsymbol{H}} = \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H} - \boldsymbol{W}^T\boldsymbol{X}$$

- Classical solution

$$\boldsymbol{H} = \boldsymbol{H} + \eta \odot (\boldsymbol{W}^T\boldsymbol{X} - \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H})$$

- Exponentiated gradient

$$\boldsymbol{H} = \boldsymbol{H} \odot e^{\eta \odot (\boldsymbol{W}^T\boldsymbol{X} - \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H})}$$

Gradient Descent

- Compute the derivative and find its zero

$$\frac{\partial E}{\partial \boldsymbol{W}} = \boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T - \boldsymbol{X}\boldsymbol{H}^T$$
$$\frac{\partial E}{\partial \boldsymbol{H}} = \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H} - \boldsymbol{W}^T\boldsymbol{X}$$

- Classical solution

$$\boldsymbol{H} = \boldsymbol{H} + \eta \odot (\boldsymbol{W}^T\boldsymbol{X} - \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H})$$

- Exponentiated gradient

$$\boldsymbol{H} = \boldsymbol{H} \odot e^{\eta \odot (\boldsymbol{W}^T\boldsymbol{X} - \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H})}$$

Multiplicative Updates

- Setting the learning rate:

$$\eta = \frac{\boldsymbol{H}}{\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H}}$$

- Results in updates:

$$\boldsymbol{W} = \boldsymbol{W} \odot \frac{\boldsymbol{X}\boldsymbol{H}^T}{\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T},$$

$$\boldsymbol{H} = \boldsymbol{H} \odot \frac{\boldsymbol{W}^T\boldsymbol{X}}{\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H}},$$

- Advantages:
  - automatic non-negativity constraint satisfaction
  - adaptive learning rate
  - no parameter setting

Multiplicative Updates

- Setting the learning rate:

$$\eta = \frac{\boldsymbol{H}}{\boldsymbol{W}^T \boldsymbol{W} \boldsymbol{H}}$$

- Results in updates:

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{W} \odot \frac{\boldsymbol{X}\boldsymbol{H}^T}{\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T}, \\ \boldsymbol{H} &= \boldsymbol{H} \odot \frac{\boldsymbol{W}^T \boldsymbol{X}}{\boldsymbol{W}^T \boldsymbol{W} \boldsymbol{H}}, \end{aligned}$$

- Advantages:
  - automatic non-negativity constraint satisfaction
  - adaptive learning rate
  - no parameter setting

Multiplicative Updates

- Setting the learning rate:

$$\eta = \frac{\boldsymbol{H}}{\boldsymbol{W}^T \boldsymbol{W} \boldsymbol{H}}$$

- Results in updates:

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{W} \odot \frac{\boldsymbol{X} \boldsymbol{H}^T}{\boldsymbol{W} \boldsymbol{H} \boldsymbol{H}^T}, \\ \boldsymbol{H} &= \boldsymbol{H} \odot \frac{\boldsymbol{W}^T \boldsymbol{X}}{\boldsymbol{W}^T \boldsymbol{W} \boldsymbol{H}}, \end{aligned}$$

- Advantages:
  - automatic non-negativity constraint satisfaction
  - adaptive learning rate
  - no parameter setting

Non-negative Quadratic Programming

$$F(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{A}\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha},$$

subject to $\alpha_i \geq 0, i \in \{1..n\}$

Outline

SVM as NMF-type problem

- SVM dual formulation

$$S(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i$$

subject to $\alpha_i \geq 0, i \in \{1..n\}$,

- Looks like NMF:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\Phi(\boldsymbol{X}_A)\alpha_A - \Phi(\boldsymbol{X}_B)\alpha_B\|_2^2 - \sum_{i\in\{A,B\}} \alpha_i$$

subject to $\alpha_i \geq 0$,

- NMF objective function:

$$E = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2$$

SVM as NMF-type problem

- SVM dual formulation

$$S(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n}\alpha_i$$

subject to $\alpha_i \geq 0, i \in \{1..n\}$,

- Rewrite the square

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\left(\sum_{ij\in A}\alpha_i\alpha_j k_{ij} - 2\sum_{\substack{i\in B\\ j\in A}}\alpha_i\alpha_j k_{ij} + \sum_{ij\in B}\alpha_i\alpha_j k_{ij}\right) - \sum_{i=1}^{n}\alpha_i$$

- Looks like NMF:

$$\min_{\alpha} \frac{1}{2}\|\Phi(\boldsymbol{X}_A)\alpha_A - \Phi(\boldsymbol{X}_B)\alpha_B\|_2^2 - \sum_{i\in\{A,B\}}\alpha_i$$

subject to $\alpha_i \geq 0$,

- NMF objective function:

SVM as NMF-type problem

- Looks like NMF:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\Phi(\boldsymbol{X}_A)\boldsymbol{\alpha}_A - \Phi(\boldsymbol{X}_B)\boldsymbol{\alpha}_B\|_2^2 - \sum_{i\in\{A,B\}} \alpha_i$$

$$\text{subject to } \alpha_i \geq 0,$$

- NMF objective function:

$$E = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2$$

SVM as NMF-type problem

- Looks like NMF:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \|\Phi(\boldsymbol{X}_A)\boldsymbol{\alpha}_A - \Phi(\boldsymbol{X}_B)\boldsymbol{\alpha}_B\|_2^2 - \sum_{i \in \{A,B\}} \alpha_i$$

$$\text{subject to } \alpha_i \geq 0,$$

- NMF objective function:

$$E = \frac{1}{2} \|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2$$

Multiplicative updates

- Differentiate the objective

$$\begin{aligned}\frac{\partial S}{\partial \boldsymbol{\alpha}_A} &= \langle \Phi(\boldsymbol{X}_A), \Phi(\boldsymbol{X}_A)\rangle \boldsymbol{\alpha}_A - \langle \Phi(\boldsymbol{X}_A), \Phi(\boldsymbol{X}_B)\rangle \boldsymbol{\alpha}_B - \mathbf{1} \\ &= K(\boldsymbol{X}_A, \boldsymbol{X}_A)\boldsymbol{\alpha}_A - (K(\boldsymbol{X}_A, \boldsymbol{X}_B)\boldsymbol{\alpha}_B + \mathbf{1})\end{aligned}$$

- Simple multiplicative updates for SVM

$$\begin{aligned}\boldsymbol{\alpha}_A &= \boldsymbol{\alpha}_A \odot \frac{K(\boldsymbol{X}_A, \boldsymbol{X}_B)\boldsymbol{\alpha}_B + \mathbf{1}}{K(\boldsymbol{X}_A, \boldsymbol{X}_A)\boldsymbol{\alpha}_A} \\ \boldsymbol{\alpha}_B &= \boldsymbol{\alpha}_B \odot \frac{K(\boldsymbol{X}_B, \boldsymbol{X}_A)\boldsymbol{\alpha}_A + \mathbf{1}}{K(\boldsymbol{X}_B, \boldsymbol{X}_B)\boldsymbol{\alpha}_B},\end{aligned}$$

Multiplicative updates (cont.)

- Multiplicative Updates for Sign Insensitive Kernel SVM

$$\boldsymbol{\alpha}_A = \boldsymbol{\alpha}_A \odot \frac{\boldsymbol{K}_{AB}^{+}\boldsymbol{\alpha}_B + \boldsymbol{K}_{A}^{-}\boldsymbol{\alpha}_A + \mathbf{1} + \boldsymbol{D}_A\boldsymbol{\alpha}_A}{\boldsymbol{K}_{A}\boldsymbol{\alpha}_A + \boldsymbol{K}_{AB}^{-}\boldsymbol{\alpha}_B + \boldsymbol{D}_A\boldsymbol{\alpha}_A}$$

$$\boldsymbol{\alpha}_B = \boldsymbol{\alpha}_B \odot \frac{\boldsymbol{K}_{BA}^{+}\boldsymbol{\alpha}_A + \boldsymbol{K}_{B}^{-}\boldsymbol{\alpha}_B + \mathbf{1} + \boldsymbol{D}_B\boldsymbol{\alpha}_B}{\boldsymbol{K}_{B}^{+}\boldsymbol{\alpha}_B + \boldsymbol{K}_{BA}^{-}\boldsymbol{\alpha}_A + \boldsymbol{D}_B\boldsymbol{\alpha}_B}$$

- semiNMF-type SVM updates

$$\boldsymbol{\alpha}_A = \boldsymbol{\alpha}_A \odot \sqrt{\frac{\boldsymbol{K}_{AB}^{+}\boldsymbol{\alpha}_B + \boldsymbol{K}_{A}^{-}\boldsymbol{\alpha}_A + \mathbf{1}}{\boldsymbol{K}_{A}^{+}\boldsymbol{\alpha}_A + \boldsymbol{K}_{AB}^{-}\boldsymbol{\alpha}_B}}$$

$$\boldsymbol{\alpha}_B = \boldsymbol{\alpha}_B \odot \sqrt{\frac{\boldsymbol{K}_{BA}^{+}\boldsymbol{\alpha}_A + \boldsymbol{K}_{B}^{-}\boldsymbol{\alpha}_B + \mathbf{1}}{\boldsymbol{K}_{B}^{+}\boldsymbol{\alpha}_B + \boldsymbol{K}_{BA}^{-}\boldsymbol{\alpha}_A}}$$

Sum Constraint and Box Constraint

- Soft Margin SVM (the box constraint)

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i$$
$$\text{subject to } 0 \leq \alpha_i \leq l, i \in \{1..n\}.$$

- Bias
  - Introduce λ and rewrite $\sum_i y_i \alpha_i = 0$ as:

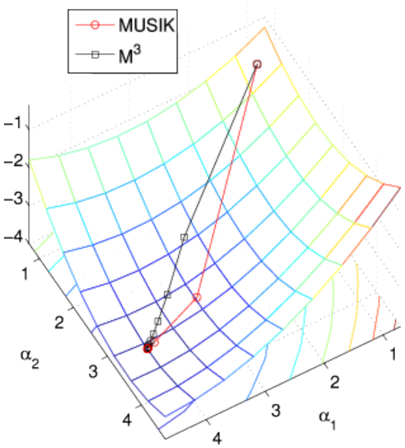
$$\sum_{i \in A} \alpha_i = \lambda, \sum_{i \in B} \alpha_j = \lambda$$

  - Redefine $\beta_k = \alpha_k / \lambda$ and solve resulting SVM for λ and $\boldsymbol{\beta}$ using multiplicative updates

Outline

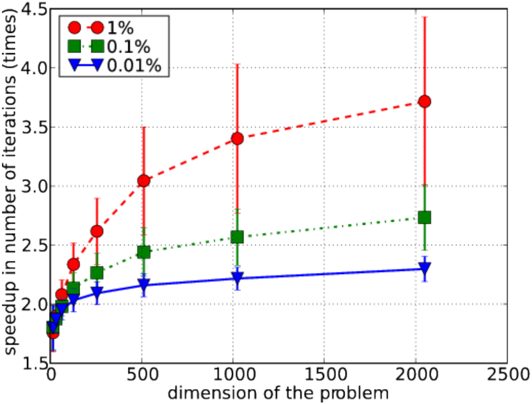
Simulations

- MUSIK takes bigger steps and follows a different path
- Converges faster within a given tolerance



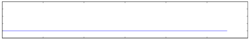
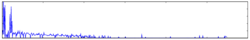
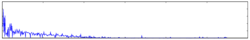
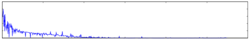
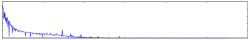
Simulations

- MUSIK takes bigger steps and follows a different path
- Converges faster within a given tolerance



it works correctly

- Check on UCI dataset Newman and Merz [1998]
- Sonar and Breast cancer data
- Convergence is fast (breast cancer dataset)

i	support vectors	$\epsilon_t(\%)$	$\epsilon_g(\%)$
0		3.8	0.0
1		2.5	3.0
2		1.5	1.5
4		0.5	1.5
8		0.2	2.3
16		0.0	2.3
64		0.0	2.3

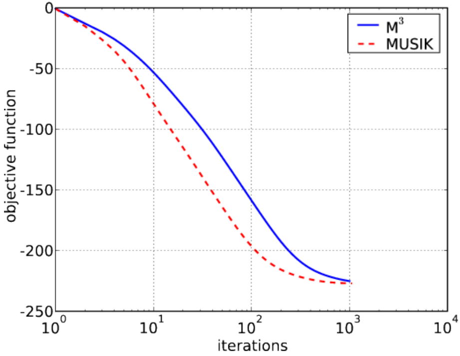
it works correctly (cont.)

Kernel		Breast			Sonar		
		M^3	M	KA	M^3	M	KA
Poly	4	2.26	2.26	2.26	9.62	9.62	9.62
Poly	6	3.76	3.76	3.76	10.58	10.58	10.58
Gaussian	3	2.26	2.26	2.26	11.53	11.53	11.53
Gaussian	1	0.75	0.75	0.75	7.69	7.69	7.69

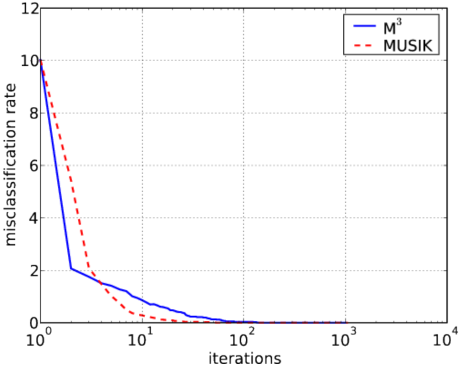
Converges to the exact same global solution

it works fast

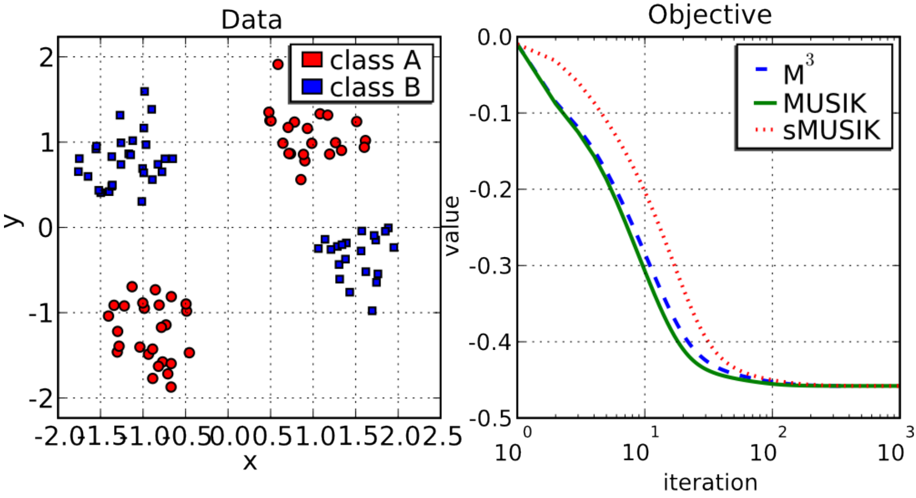
Objective



Training error



... and it is general



Conclusions

- Clean connection between SVM and NMF
- Fully multiplicative algorithm for SVM
- Simple to code algorithm: about 5 Matlab lines
- Speed Improvements
  Theoretically (asymptotic convergence rates) and practically faster
- Possibility for algorithm reuse
  - SVM for NMF
  - NMF for SVM
- Further details in SDM 2009 paper

Thank you!

Bibliography

C. L. Blake D. J. Newman and C. J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html`.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, 2001. ISBN 0262194759. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/0262194759`.