Reading Club: Support Vector Machines

Scalability and Multi-Class Methods

Outline

- About the Scalability of SVMs
  - Quadratic Programming Problem
  - Interior-Point-Method
  - Generalisation Performance
  - Solutions – An overview
- Methods for Multi-class SVMs
  - Split into several problems:
    - One-Against-All
    - One-Against-One
    - DAG-Methods
  - All-Together:
    - Weston & Watkins
    - Crammer & Singer

About the Scalability of SVMs

Quadratic Programming Problem

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

$$\text{subject to} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, n,$$

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

- Non-Linear Quadratic Program
  - Inequality constraints prohibit the transformation into a linear program
- The good news:
  - Size of the optimization problem: Independent of dimensions of input & feature spaces
- But: Quadratic problems are still computationally expensive
  - Worst Case: NP-Hard, but mostly $O(N^3)$ or even polynomial

Quadratic Programming Problem

$$\max_{\alpha} \quad -\frac{1}{2}\sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j k\left(\mathbf{x}_i, \mathbf{x}_j\right) + \sum_{i=1}^{N} \alpha_i$$

$$\text{s.t.} \sum_{i=1}^{N} \alpha_i y_i = 0,$$

$$0 \leq \alpha_i \leq C, \ \forall i \in \{1, \ldots, N\}.$$

Can be rewritten to:

$$\begin{aligned} &\text{minimize:} && W(\boldsymbol{\alpha}) = -\boldsymbol{\alpha}^T + \frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{\alpha}^{\mathbf{T}}\mathbf{Q}\boldsymbol{\alpha} \\ &\text{subject to:} && \boldsymbol{\alpha}^T\mathbf{y} = \mathbf{0} \\ & && \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{C} \end{aligned}$$

Where:

$$Q \text{ as } (Q)_{ij} = y_i y_j k(\mathbf{x_i}, \mathbf{x_j})$$

$\boldsymbol{\alpha}$ is a vector of ℓ variables

Quadratic Programming Problem

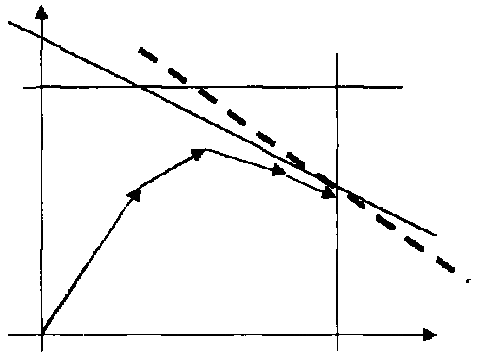
$$\begin{aligned} &\text{minimize:} && W(\boldsymbol{\alpha}) = -\boldsymbol{\alpha}^T + \frac{1}{2}\boldsymbol{\alpha}^{\mathbf{T}}\mathbf{Q}\boldsymbol{\alpha} && Q \text{ as } (Q)_{ij} = y_i y_j k(\mathbf{x_i}, \mathbf{x_j}) \\ &\text{subject to:} && \boldsymbol{\alpha}^T \mathbf{y} = \mathbf{0} && \boldsymbol{\alpha} \text{ is a vector of } \ell \text{ variables} \\ & && \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{C} \end{aligned}$$

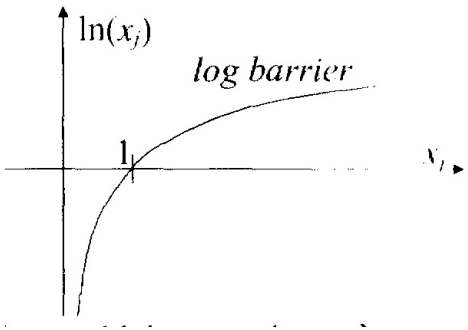
- Positive semi-definite matrix Q is of size N^2
  - Memory requirement for storage of Q (double precision): N^2 * 8byte
- "out-of-core" algorithms (virtual memory)
  - Caching: reduce input-ouput-operations
- Recomputation of Q
  - transform memory bottleneck into computational bottleneck (computational costs depending on Kernel complexity!)
  - Prohibitive when Q is needed often

Interior-Point (Barrier) Method

- Solves linear problems very efficiently
  - at the very most polynomial
- Iteratively follow a trajectory in the interior of the admissible region
  - Number of iterations depends on size of the optimization problem, hence *N!*
- For quadratic problems
  - In each iteration a linear program (size depends on *N!*) has to be solved
- For large *N:*
  - Computational and Memory capacities reach their limits
  - Recomputation of *Q* infeasible (esp. for complex Kernels!)

Visualization for LP:





Generalization Performance

- Bound on the Expected Risk strongly depends on number of examples *N*

$$I[f] \leq I_{emp}[f, \mathcal{D}] + \frac{4\kappa^2 B^2 \gamma}{N} + \left(8\kappa^2 B^2 \gamma + 2B\right) \sqrt{\frac{\ln 1/\eta}{2N}},$$

- Trade-Off:
  - Large *N* required for good generalisation performance!
  - Memory and computational costs are prohibitively high for large *N!*

Solutions – An Overview

- Decomposition
  - Split quadratic problem into a series of smaller tasks of fixed size
    - Inactive and active part (working set)
  - Working set selection strategies
    (Zoutendijk‘s method: first-order approximation to the target function)
  - Memory requirements: linear in *N* and in the number of support vectors
  - BUT: Long training times!

Solutions – An Overview

- Chunking
  - Similar to decomposition, but varying problem size
  - Exploits the property of SVM having few support vectors
  - Converges to a problem of size equal to the number of support vectors
  - At each Step:
    - Solve a subproblem for all non-zero α_i & some α_i violating the KKT conditions

Solutions – An Overview

- Shrinking
  - Shrinks the optimization problem successively eliminating examples
  - Lagrange multipliers indicate variables that will end up at a bound constraint
    - Fix corresponding data points (no further computation)
  - Heuristic! Check optimality conditions after convergence (Reoptimization)
  - Additionally, caching-strategies can be used very efficiently

Solutions – An Overview

- Sequential Minimal Optimization (SMO)
  - Most extrem case of decomposition: In each iteration solve a QP of size two
  - Analytically solvable -> no QP optimizer necessary!
  - Heuristics for selection of the two data points is critical!

Solutions – An Overview

- Other techniques:
  - LS-SVM (Avoiding QP, BUT: Sparsity is lost -> Classification: O(N))
  - Sparse LS-SVM
  - Boost-SMO
  - Kernel-Adatron
    - Based on online learning (Does not allow for training errors!)
  - Likelyhood-based squashing with a probabilistic formulation of SVMs
  - Reduced set method (approximating the SV decision surface)
  - For the role of Kernel-Selection on computational and memory costs please refer to Hamers 2004

Methods for Multi-class SVMs

Overview

- Two types of approaches:
  - Constructing and combining several binary classifiers
    - One-against-all
    - One-against-one
    - Directed Acyclic Graph (DAG)
  - Directly consider all classes in one optimization problem (all-together)
    - *k*-class SVM by Weston & Watkins
    - *k*-class SVM by Crammer & Singer
- Naming conventions:
  - Indices:
    - Classes: $i = 1,..,k$
    - Examples: $n = 1,..,l$

One-Against-All-Method

- Train k SVM models:
  - For the ith SVM class i is positive all others are negative:
    - k l-variable quadratic programming problems
  - Decision functions:
    - Class of $x = \text{argmax}_i ((w^i)^T \Phi(x) + b^i)$
    - Choose the class with the largest value

    (Indicating the distance to the dicision function!)
  - Training time scales linearly with k
  - No bound on the generalization error for the single SVMs

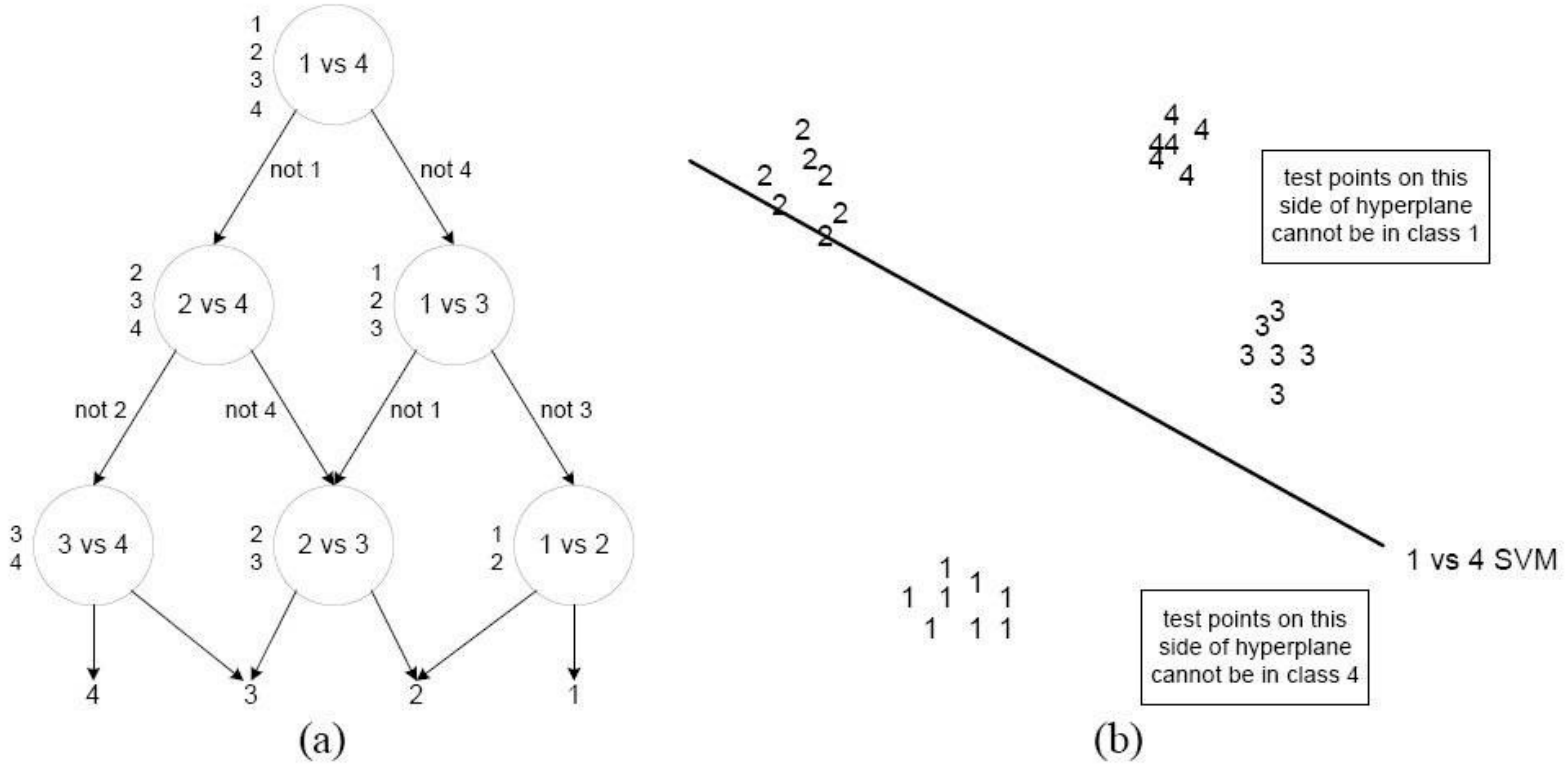
One-Against-One-Method

- Train *k(k-1)/2* SVM models:
  - Collect all data from the *ith* and a second class $j \neq i$
  - Train a classifier on each combination:
    - *k(k-1)/2* quadratic programming problems of varying size
    - Size is in average *2*l/k*
  - Decision function:
    - Different combination strategies possbile
    - „Max Wins“ (has no bound on generalization error -> tends to overfit):
      - if (sign$((w^{ij})^T \Phi(x) + b^{ij}))$ *then vote(i)++* *else vote(j)++*
      - Class of x = argmax_i vote(i)

DAG-Method

- Training exactly as one-against-one
- But different combination strategy in testing phase:
  - Rooted binary directed acyclic graph
    - *k(k-1)/2* internal nodes (binary SVMs)
    - *k* leaves
  - Performance of the Graph depends on:
    - Size of the graph and
    - margins achieved at the nodes

DAG-Method


1 2 3 4
1 vs 4
not 1
not 4
2 3 4
2 vs 4
1 2 3
1 vs 3
not 2
not 4
not 1
not 3
3 4
3 vs 4
2 3
2 vs 3
1 2
1 vs 2
4
3
2
1
(a)
test points on this side of hyperplane cannot be in class 1
1 vs 4 SVM
test points on this side of hyperplane cannot be in class 4
(b)

All-Together-Method: Weston et al

- *k*-class SVM by Weston & Watkins:
- One single optimization problem

$$\min_{w,b,\xi} \quad \frac{1}{2}\sum_{i=1}^{k} w_i^T w_i + C\sum_{n=1}^{l}\sum_{i\neq y_n} \xi_n^i$$

$$w_{y_n}^T \phi(x_n) + b_{y_n} \geq w_i^T \phi(x_n) + b_i + 2 - \xi_n^i,$$

$$\xi_n^i \geq 0, n = 1, \ldots, l, \quad i \in \{1, \ldots, k\} \backslash y_n$$

- *k* decision functions but all obtained by the problem above
  - Class of $x = \text{argmax}_i\ ((w^i)^T\ \Phi(x) + b^i)$

All-Together-Method: Weston et al

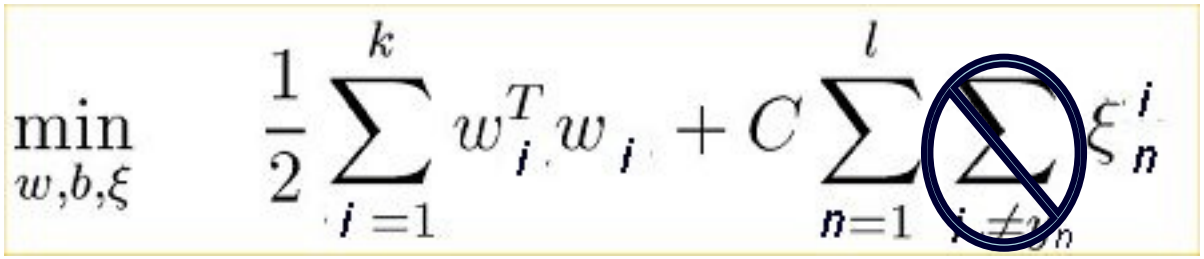
- The dual formulation contains *kl* variables where *l* of them are always zero
  - -> *(k-1)l* variables
  - -> Impractical for large problems!
- Decomposition (working set) and caching methods necessary!
- Instead of a single linear constraint, we now have *k* linear constraints
- Attention! SMO (2 variables in working set) not applicable:
  - If (#variables < *k*) then it may be an over-determined linear system (more equations than variables)
    - -> solution of the decomposition algorithm cannot be moved

No working set selection strategy known, that ensures a decrease of the objective function! (Because of bounded constraints)

A trick can overcome this issue, but accuracy might be affected

All-Together-Method: Crammer et al

- *k*-class SVM by Crammer & Singer:



  - Main difference: Use only *l* slack variables
  - Working set selections for the decomposition methods can be directly conducted
  - This was only a short summary! For more details see Hsu & Lin
  - *See also: „k*-class linear programming machines“ in Watkins et al. and other methods

Literature

General literature:

- *Hamers, Bart: „Kernel Models for Large Scale Applications“,* Katholieke Universiteit Leuven – Faculteit ToegepasteWetenschappen, Heverlee (Belgium), 2004
- Joachims, Thorsten: *„Making Large-Scale SVM Learning Practical“,* IN: Schölkopf et. al.: ‚Advances in Kernel Methods – Support Vector Learning‘, MIT Press, Cambridge, USA, 1999
- Müller et al.: *„An Introduction to Kernel-Based Learning Algorithms“*, IEEE Transactions on Neural Networks, Vol. 12, No. 2, March 2001
- Hsu & Lin: *„A Comparison of Methods for Multi-class Support Vector Machines*”

Literature

More specialised literature:

- Burges et al.: *„Improving the Accuracy and Speed of Support Vector Machines”*, In: Mozer et al.: ‘Neural Information Processing Systems’, Vol. 9, MIT Press, Cambridge, 1997
- Chang et al.: *„The Analysis of Decomposition Methods for Support Vector Machines*”*
- Frieß et al.: *„The Kernel-Adatron Algorithm: a Fast and Simple Learning Procedure for Support Vector Machines”*
- Pavlov et al.: *„Scaling-up Support Vector Machines Using Boosting Algorithm”*
- Pavlov et al.: *„Towards Scalable Support Vector Machines using Squashing”*
- Platt: *„Fast Training of Support Vector Machines using Sequential Minimal Optimization”,* IN: Schölkopf et. al.: ‚Advances in Kernel Methods – Support Vector Learning‘, MIT Press, Cambridge, USA, 1999
- Suykens et al.: *„Least Squares Support Vector Machine Classifers”*, Kluwer Academic Publishers, Netherlands, 1998

Literature

More specialised literature, cont'd:

- Suykens et al.: *„Least Squares Support Vector Machine Classifers: a Large Scale Algorithm"*
- Suykens et al.: *„Sparse Least Squares Support Vector Machine Classifers"*
- Osuna et al.: *„Training support vector machines: An application to face detection"*, IN: editor, Proceedings, CVPR'97
- Vapnik: *„Estimation of Dependences Based on Empirical Data"*, Springer Verlag, Berlin, 1982
- Zoutendijk, G.: *„Methods of Feasible Directions: a Study in Linear and Non-linear Programming"*, Elsevier, 1970
- http://en.wikipedia.org/wiki/Quadratic_programming
- http://svr-www.eng.cam.ac.uk/~kkc21/thesis_main/node4.html
- Weston & Watkins: *„Multi-class Support Vector Machines", 1998*
- *Platt et al.: „Large Margin DAGs for Multiclass Classification"*