COM S 778

A Support Vector Method for Multivariate Performance Measures

by Thorsten Joachims
International Conference on Machine Learning 2005

Presented by: Thorsten Joachims

Thanks to
Rich Caruana, Alexandru Niculescu-Mizil, Pierre Dupont, Jérôme Callut

Supervised Learning

- **Find function from input space *X* to output space *Y***

$$h : X \longrightarrow \{+1, -1\}$$

such that the ~~prediction error~~ is low.

Text Classification:
- F_1-Score
- Precision/Recall Break-Even (PRBEP)

Medical Diagnosis:
- ROC Area

Information Retrieval:
- Precision at 10

Related Work

- **Approach "Estimate Probabilities"**
  - E.g. [Platt, 2000] [Langford & Zadrozny, 2005] [Niculescu-Mizil & Caruana, 2005]
  - Potentially solve harder problem than required
- **Approach "Optimize Substitute Loss, then Post-Process"**
  - E.g. [Lewis, 2001] [Yang, 2001] [Abe et al. 2004] [Caruana & Niculescu-Mizil, 2004]
  - Typically multi-step approach, cross-validation
- **Approach "Directly Optimize Desired Loss"**
  - Linear cost models: e.g. [Morik et al., 1999] [Lin et al., 2002]
  - ROC-Area: e.g. [Herbrich et al. 2000] [Rakotomamonjy, 2004] [Cortes & Mohri, 2003] [Freund et al., 1998] [Yan et al., 2003] [Ferri et al., 2002]
  - F_1-Score: difficult [Musicant et al. 2003]

Overview

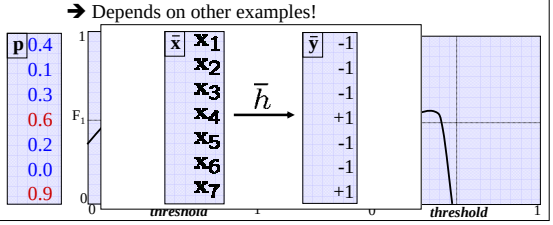
- **Formulation of Support Vector Machine for**
  - any loss function that can be computed from the contingency table.
    - F1-score, Error Rate, Linear Cost Models, etc.
  - any loss function that can be computed from contingency tables with cardinality constraints.
    - PRBEP, Prec@k, Rec@k, etc.
  - ROC-Area
- **Polynomial Time Algorithm**
- **Conventional classification SVM is special case**
  - New optimization problem
  - New representation and (extremely sparse) support vectors

Optimizing F_1 -Score

- ***F_1* -score is non-linear function of example set**
  - F_1-score: harmonic average of precision and recall

$$F_1 = \frac{2\,Prec\,Rec}{Prec + Rec}$$

  - For example vector $\mathbf{x_1}$. Predict $y_1=1$, if $P(y_1=1|\mathbf{x_1})=0.4$?
    ➔ Depends on other examples!



Approach: Multivariate Prediction

- **Training Data:** $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)) \sim_{i.i.d} \Pr(\mathbf{x}, y)$
- **Conventional Setting:** learn $h : X \longrightarrow \{-1, +1\}$

$$R^{\delta}(h) = \int \delta\left(h(\mathbf{x}'), y'\right) d\Pr(\mathbf{x}', y')$$

$$\hat{R}^{\delta}_{S}(h) = \frac{1}{n}\sum_{i=1}^{n} \delta\left(h(\mathbf{x}_i), y_i\right)$$

- **Multivariate Setting:** learn $\bar{h} : X^n \longrightarrow \{-1, +1\}^n$

$$R^{\Delta}(\bar{h}) = \int \Delta\left(\bar{h}(\mathbf{x}'_1, \ldots, \mathbf{x}'_{n'}), (y'_1, \ldots, y'_{n'})\right) d\Pr(S')$$

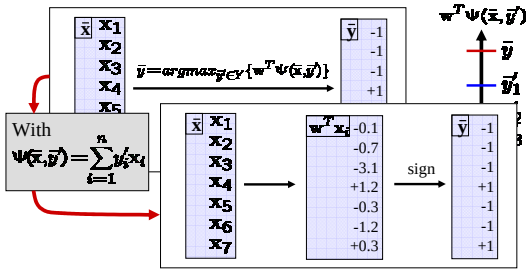
$$\hat{R}^{\Delta}_{S}(\bar{h}) = \Delta\left(\bar{h}(\mathbf{x}_1, \ldots, \mathbf{x}_n), (y_1, \ldots, y_n)\right)$$

Note:
If $\Delta\left(\bar{h}(\mathbf{x}_1, \ldots, \mathbf{x}_n), (y_1, \ldots, y_n)\right) = \frac{1}{n}\sum_{i=1}^{n} \delta\left(h(\mathbf{x}_i), y_i\right)$
then both settings are equivalent.

Multivariate Support Vector Machine

Approach: Linear Discriminant [Collins 2002] [Lafferty et al. 2002] [Taskar et al. 2004] [Tsochantaridis et al. 2004] etc.

- "Learn weights $\mathbf{w}$ so that $\mathbf{w}^T\Psi(\vec{x},\vec{y})$ is max for correct $\vec{y}$."



Multivariate SVM Optimization Problem

Approach: Structural SVM [Taskar et al. 04] [Tsochantaridis et al. 04]

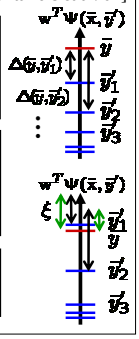
Hard-margin optimization problem:

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$s.t.\ \forall \vec{y}'\in\vec{\mathcal{Y}}\setminus\{\vec{y}\} : \mathbf{w}^T\Psi(\vec{x},\vec{y}) \geq \mathbf{w}^T\Psi(\vec{x},\vec{y}') + 1$$

Soft-margin optimization problem:

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$$
$$s.t.\ \forall \vec{y}'\in\vec{\mathcal{Y}} : \mathbf{w}^T\Psi(\vec{x},\vec{y}) \geq \mathbf{w}^T\Psi(\vec{x},\vec{y}') + \Delta(\vec{y},\vec{y}') - \xi$$

Theorem: At the solution, the training loss is upper bounded by $\Delta(\vec{y},\vec{h}(\vec{x})) \leq \xi$.



Multivariate SVM Generalizes Classification SVM

Theorem: The solutions of the multivariate SVM with number of errors as the loss function and an (unbiased) classification SVM are equal.

Multivariate SVM optimizing Error Rate:

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$$
$$s.t.\ \forall \vec{y}'\in\vec{\mathcal{Y}} : \mathbf{w}^T\Psi(\vec{x},\vec{y}) \geq \mathbf{w}^T\Psi(\vec{x},\vec{y}') + 2Err(\vec{y},\vec{y}') - \xi$$

$\Leftrightarrow$

Classification SVM (unbiased):

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + 2C\sum_{i=1}^{n}\xi_i$$
$$s.t.\ \ y_1(\mathbf{w}^T\mathbf{x}_1) \geq 1-\xi_1,\ \ \ldots\ ,\ \ y_n(\mathbf{w}^T\mathbf{x}_n) \geq 1-\xi_n$$

Sparse Approximation Algorithm for Multivariate SVM

Approach: Sparse Approx. Structural SVM [Tsochantaridis et al. 04]

- **Input:** $\vec{\mathbf{x}} = (\mathbf{x}_1,\ldots,\mathbf{x}_n), \vec{y} = (y_1,\ldots,y_n), C, \epsilon$
- $S \leftarrow \emptyset, \mathbf{w} \leftarrow 0, \xi \leftarrow 0$
- **REPEAT**
  - compute $\vec{y}' = argmax_{\vec{y}\in Y}\{\Delta(\vec{y},\vec{y}') + \mathbf{w}^T\Psi(\vec{x},\vec{y}')\}$
  - IF $(\Delta(\vec{y},\vec{y}') - \mathbf{w}^T[\Psi(\vec{x},\vec{y}) - \Psi(\vec{x},\vec{y}')]) > \xi + \epsilon$
    - $S \leftarrow S \cup \{\mathbf{w}^T[\Psi(\vec{x},\vec{y}) - \Psi(\vec{x},\vec{y}')] \geq \Delta(\vec{y},\vec{y}') - \xi\}$
    - $[\mathbf{w},\xi] \leftarrow$ optimize SVM objective over S
  - ENDIF
- **UNTIL** S **has not changed during iteration**

Find most violated constraint

Violated by more than ε ?

Add constraint to working set

Polynomial Convergence Bound

- **Theorem [Tsochantaridis et al., 2004]:** The sparse-approximation algorithm finds a solution to the soft-margin optimization problem after adding at most

$$\max\left\{\frac{2L}{\epsilon}, \frac{8Cn^2R^2L}{\epsilon^2}\right\}$$

constraints to the working set S, so that the Kuhn-Tucker conditions are fulfilled up to a precision ϵ. The loss has to be bounded $0 \leq \Delta(\vec{y},\vec{y}') \leq L$, and $R = \max_i \|\mathbf{x}_i\|$.

ARGMAX for Contingency Table

- **Problem:**
  - $\underset{\vec{y}\in\{+1,-1\}^n}{\operatorname{argmax}} \{\Delta(\vec{y},\vec{y}') + \mathbf{w}^T\sum_{i=1}^{n} y_i\mathbf{x}_i\}$
- **Key Insight:**
  - Only n^2 different contingency tables exist.
  - ARGMAX for each table easy to compute via sorting.
  - Time $O(n^2)$
- **Applies to:**
  - Errorrate, F_1, Prec@k, Rec@k, PRBEP, etc.

```
1: Input: x=(x_1,...,x_n), y=(y_1,...,y_n), y'
2: (c_1^+,...,c_#pos^+) ← sort{i : y_i=+1} by w^T x_i
3: (c_1^-,...,c_#neg^-) ← sort{i : y_i=-1} by w^T x_i
4: for a ∈ [0,...,#pos] do
5:    c ← #pos − a
6:    set y'_{c_1^+},...,y'_{c_a^+} to 1
7:    set y'_{c_{a+1}^+},...,y'_{c_#pos^+} to −1
8:    for d ∈ [0,...,#neg] do
9:       b ← #neg − d
10:      set y'_{c_1^-},...,y'_{c_b^-} to 1
11:      set y'_{c_{b+1}^-},...,y'_{c_#neg^-} to −1
12:      v ← Δ(a,b,c,d) + w^T Σ_{i=1}^n y'_i x_i
13:      if v is the largest so far then
14:         y* ← (y'_1,...,y'_n)
15:      end if
16:   end for
17: end for
18: return(y*)
```

ARGMAX for ROC-Area

- **Problem:**
  - $\text{argmax}_{\vec{y}\in\{1,-1\}^n} \{\Delta(\vec{y},\vec{y}) + \mathbf{w}^T \sum_{i=1}^{n} y_i \mathbf{x}_i\}$
- **Key Insight:**
  - ROC Area is proportional to "swapped pairs"
  - Loss decomposes linearly over pairs
  - Find argmax via sort in time $O(n \log n)$
  - Represent n^2 pairs as

$$\Psi(\vec{x},\vec{y}) = \sum_{i=1}^{n} c_i \mathbf{x}_i \text{ with } c_i = \begin{cases} \sum_{j=1}^{\#neg} y_{ij}, & if(y_i=1) \\ \sum_{j=1}^{\#pos} y_{ij}, & if(y_i=-1) \end{cases}$$

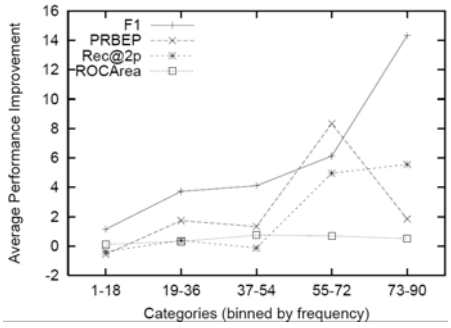
```
1: Input: x̄ = (x_1,...,x_n), ȳ = (y_1,...,y_n)
2: for i∈{i: y_i=1} do s_i ← −0.25+w^T x_i
3: for i∈{i: y_i=−1} do s_i ← 0.25+w^T x_i
4: (r_1,...,r_n) ← sort {1,...,n} by s_i
5: s_p = #pos, s_n = 0
6: for i ∈ {1,...,n} do
7:   if y_{r_i} > 0 then
8:     c_{r_i} ← (#neg − 2 s_n)
9:     s_p ← s_p − 1
10:  else
11:    c_{r_i} ← (−#pos + 2 s_p)
12:    s_n ← s_n + 1
13:  end if
14: end for
15: return(c_1,...,c_n)
```

Experiment: Generalization Performance

- **Experiment Setup**
  - Macro-average over all classes in dataset
  - Baseline: classification SVM with linear cost model
  - Select C and cost ratio j via 2/3 – 1/3 holdout test
  - Two-tailed Wilcoxon (**=95%, *=90%)

Dataset	Method	F_1	PRBEP	$Rec_{@2p}$	ROCArea
Reuters (90 classes)	SVM^{Δ}_{multi}	62.0	68.2	78.3	99.1
Examples: 9603/3299	SVM_{org}	56.1	65.7	77.2	98.6
Features: 27658	win/lose	(51/20)**	(16/8)**	(14/8)	(43/33)*
ArXiv (14 classes)	SVM^{Δ}_{multi}	56.8	58.4	73.3	92.8
Examples: 1168/32487	SVM_{org}	49.6	57.9	74.4	92.7
Features: 13525	win/lose	(9/5)*	(9/4)	(1/13)**	(8/6)
Optdigits (10 classes)	SVM^{Δ}_{multi}	92.5	92.7	98.4	99.4
Examples: 3823/1797	SVM_{org}	91.5	91.5	98.7	99.4
Features: 64	win/lose	(8/2)*	(5/1)*	(1/5)	(6/4)
Covertype (7 classes)	SVM^{Δ}_{multi}	73.8	72.1	93.1	94.6
Examples: 1000/2000	SVM_{org}	73.9	71.0	94.7	94.1
Features: 54	win/lose	(3/4)	(5/2)	(2/5)	(4/3)

Experiment: Unbalanced Classes in Reuters



Experiment: Number of SV

Corollary: For error rate as the loss function, the hard-margin solution after the first iteration is equal to Rocchio Algorithm.

$$\mathbf{w}_1 \sim \sum_{\{pos:y_{pos}=1\}} \mathbf{x}_{pos} - \sum_{\{neg:y_{neg}=-1\}} \mathbf{x}_{neg}$$

Dataset	Method	F_1	PRBEP	$Rec_{@2p}$	ROCArea	Err
Reuters (90 classes)	SVM^{Δ}_{multi}	62.0	45.0	46.3	5.1	86.3
Examples: 9603/3299	SVM_{org}					371.4
Features: 27658						
ArXiv (14 classes)	SVM^{Δ}_{multi}	129.5	43.4	45.3	26.8	177.7
Examples: 1168/32487	SVM_{org}					645.3
Features: 13525						
Optdigits (10 classes)	SVM^{Δ}_{multi}	19.6	14.6	14.0	3.9	25.0
Examples: 3823/1797	SVM_{org}					556.9
Features: 64						
Covertype (7 classes)	SVM^{Δ}_{multi}	12.5	12.0	9.4	5.0	17.1
Examples: 1000/2000	SVM_{org}					372.8
Features: 54						

Conclusions

- **Generalization of SVM to multivariate loss functions**
  - Classification SVMs are special case
- **Polynomial time training algorithms for**
  - any loss function based on contingency table.
  - ROC-Area.
- **New representation of SVM optimization problem**
  - Support Vectors represent vector of classifications
  - Can be extremely sparse
- **Future work**
  - Other performance measures, other methods (e.g. boosting)
  - Faster training algorithm exploiting special structure