

CONSISTENT ALGORITHMS FOR MULTI-LABEL CLASSIFICATION WITH MACRO-AT- k METRICS

Erik Schultheis¹ Wojtek Kotłowski² Marek Wydmuch² Rohit Babbar^{1,3} Strom Borman⁴ Krzysztof Dembczyński^{2,5}

¹Aalto University, Helsinki, Finland ²Poznan University of Technology, Poland ³University of Bath, UK ⁴Yahoo Research, Champaign, USA ⁵Yahoo Research, New York, USA

Multi-label classification with budget at k

– Multi-label classification: $\mathbf{x} \in \mathcal{X} \rightarrow \mathbf{y} \in \mathcal{Y} := \{0, 1\}^m$,
 – In areas of **extreme classification** and **recommender systems**, many problems are naturally **budgeted at k** , i.e., one needs to make exactly k predictions ($\|\hat{\mathbf{y}}\|_1 = k$).

Similar items may deliver to you quickly

Macro-at- k matrices

Instead of calculating and averaging performance **instance-wise**, calculate per-label ψ_j and average over labels (**label-wise**):

$$\Psi@k(\mathbf{Y}, \hat{\mathbf{Y}}) = \sum_{j=1}^m \psi_j(\mathbf{y}_{:j}, \hat{\mathbf{y}}_{:j}).$$

– Can balance contribution from all labels.

– **Reduces to macro-average** if all ψ_j are the same.

– Covers (instance) Precision@ k and Hamming score.

These performance measures can be written as a function of **label-wise confusion tensor**:

$$\hat{\mathbf{C}}(\mathbf{Y}, \hat{\mathbf{Y}}) := [\mathbf{C}(\mathbf{y}_{:1}, \hat{\mathbf{y}}_{:1}), \dots, \mathbf{C}(\mathbf{y}_{:m}, \hat{\mathbf{y}}_{:m})].$$

This tensor is a vector of **binary confusion matrices** for each label:

$$\hat{\mathbf{C}}(\mathbf{y}, \hat{\mathbf{y}}) := \begin{pmatrix} \text{tn} := \frac{1}{n} \sum_{i=1}^n (1 - y_i)(1 - \hat{y}_i) & \text{fp} := \frac{1}{n} \sum_{i=1}^n (1 - y_i)\hat{y}_i \\ \text{fn} := \frac{1}{n} \sum_{i=1}^n y_i(1 - \hat{y}_i) & \text{tp} := \frac{1}{n} \sum_{i=1}^n y_i\hat{y}_i \end{pmatrix}.$$

We call metrics Ψ defined on $\hat{\mathbf{C}}$ **general confusion tensor measures**.

Population Utility Framework

Two **conflicting** statistical frameworks:

Expected Test Utility (ETU): Given a **specific set of instances** with unknown stochastic labels, how well can we expect a classifier to perform on the population level, i.e., for an infinite sample:

$$\Psi_{ETU} = \mathbb{E}_{\mathbf{Y}|X} [\Psi(\hat{\mathbf{C}}(\mathbf{Y}, \hat{\mathbf{Y}}))]$$

Population Utility (PU): Given a **distribution of instances**, how well can we expect a classifier to perform on the population level, i.e., for an infinite sample:

$$\Psi_{PU} = \Psi \left(\mathbb{E}_{\mathbf{y}} [\hat{\mathbf{C}}(\mathbf{y}, \hat{\mathbf{y}})] \right).$$

The PU allows for **pointwise** predictions, independently for each x , while ETU requires whole set of instances to provide the prediction.

The optimal PU classifier

– Conditional **marginal probability** of a label: $\eta_j(\mathbf{x}) := \mathbb{P}[y_j = 1 | \mathbf{x}]$.
 – **Randomized** classifier budgeted at k is defined as probability of predicting y_j for given \mathbf{x} : $h_j(\mathbf{x}) := \mathbb{P}[\hat{y}_j = 1 | \mathbf{x}]$ and $\|\mathbf{h}\|_1 = k$
 – **Population** confusion matrix of \mathbf{h} :

$$\mathbf{C}^j(h_j) := \mathbb{E}_{\mathbf{x}} [\hat{\mathbf{C}}] = \begin{pmatrix} \mathbb{E}_{\mathbf{x}}[(1 - \eta_j(\mathbf{x}))(1 - h_j(\mathbf{x}))] & \mathbb{E}_{\mathbf{x}}[(1 - \eta_j(\mathbf{x}))h_j(\mathbf{x})] \\ \mathbb{E}_{\mathbf{x}}[\eta_j(\mathbf{x})(1 - h_j(\mathbf{x}))] & \mathbb{E}_{\mathbf{x}}[\eta_j(\mathbf{x})h_j(\mathbf{x})] \end{pmatrix}.$$

– Set of confusion tensors achievable by \mathbf{h} : $\mathcal{C}_P^{\otimes k} := \{\mathbf{C}(\mathbf{h}) : \mathbf{h} \in \mathcal{H}\}$.
 – We proof that $\mathcal{C}_P^{\otimes k}$ is compact, simplifying the subsequent analysis.

– Optimal classifier for **linear metric** $\Psi(\mathbf{C}) = \mathbf{G} \cdot \mathbf{C} = \sum_{j=1}^m \mathbf{G}^j \cdot \mathbf{C}^j$ for **gain tensor** $\mathbf{G} = (\mathbf{G}^1, \dots, \mathbf{G}^m)$ is:

$$\mathbf{h}^*(\mathbf{x}) = \text{top}_k(\mathbf{a} \odot \boldsymbol{\eta}(\mathbf{x}) + \mathbf{b}),$$

where the vectors \mathbf{a} and \mathbf{b} are given by:

$$a_j = G_{00}^j + G_{11}^j - G_{01}^j - G_{10}^j, \quad b_j = G_{01}^j - G_{00}^j.$$

– The optimal classifier for **general confusion tensor metrics**:
 – If: $\boldsymbol{\eta}(\mathbf{x})$ is **absolutely continuous**,
 – and Ψ is **strictly monotonic** and **differentiable**,
 there exists an optimal $\mathbf{C}^* \in \mathcal{C}_P^{\otimes k}$, that is $\Psi(\mathbf{C}^*) = \Psi^*$. Any classifier \mathbf{h}^* maximizing the linear utility $\mathbf{G} \cdot \mathbf{C}(\mathbf{h})$ with $\mathbf{G} = \nabla_{\mathbf{C}} \Psi(\mathbf{C}^*)$ also maximizes $\Psi(\mathbf{h})$.

Difficulty of optimizing metrics-at- k

The budget at k **couples** the label-wise binary problems:

Distribution A B	Optimal $\mathbf{h}_A^*(\mathbf{x})$ $\mathbf{h}_B^*(\mathbf{x})$			
P	π_1	π_2	π_3	
x_1	0.5	0.4	0.2	0.6
x_2	0.5	0.8	0.4	0.4 0.8

x_1	1 0	0 1	1
x_2	1	1	0

$$\Psi_{\text{jaccard}}(\mathbf{C}(\mathbf{h}_A^*, \mathbf{A})) \approx 0.454, \quad \Psi_{\text{jaccard}}(\mathbf{C}(\mathbf{h}_B^*, \mathbf{B})) \approx 0.471.$$

Finding the optimal PU classifier

We find the optimal classifier \mathbf{h} using **Frank Wolfe-based** optimization procedure:

1. Split training set \mathcal{S} into \mathcal{S}_1 (for learning $\hat{\boldsymbol{\eta}}(\mathbf{x})$) and \mathcal{S}_2 (used by Frank-Wolfe).
2. Initialize probabilistic classifier as a set of deterministic classifiers $\mathbf{h} \leftarrow \{\mathbf{h}^0\}$ and their probabilities $\alpha \leftarrow \{\alpha^0 = 1\}$ and calculate $\mathbf{C}^0 \leftarrow \hat{\mathbf{C}}(\mathbf{h}^0, \mathcal{S}_2)$.
3. Perform iteration i until convergence:
 - 3.1. Calculate tensor of gradients of \mathbf{C}^{i-1} with respect to Ψ :
 $\mathbf{G}^i \leftarrow \nabla_{\mathbf{C}} \Psi(\mathbf{C}^{i-1})$.
 - 3.2. Construct the next classifier: $\mathbf{h}^i(\mathbf{x}) \leftarrow \text{top}_k(\mathbf{a}^i \odot \hat{\boldsymbol{\eta}}(\mathbf{x}) + \mathbf{b}^i)$ with
 $\mathbf{a}^i \leftarrow \mathbf{G}_{11}^i + \mathbf{G}_{00}^i - \mathbf{G}_{01}^i - \mathbf{G}_{10}^i$, $\mathbf{b}^i \leftarrow \mathbf{G}_{01}^i - \mathbf{G}_{00}^i$.
 - 3.3. Calculate the confusion tensor of the next classifier \mathbf{h}^i : $\mathbf{C}' \leftarrow \hat{\mathbf{C}}(\mathbf{h}^i, \mathcal{S}_2)$.
 - 3.4. Find the best combination of \mathbf{C}^{i-1} and \mathbf{C}' :
 $\alpha^i \leftarrow \underset{\alpha \in [0,1]}{\operatorname{argmax}} \Psi((1 - \alpha)\mathbf{C}^{i-1} + \alpha\mathbf{C}')$,
 $\mathbf{C}^i \leftarrow (1 - \alpha^i)\mathbf{C}^{i-1} + \alpha^i\mathbf{C}'$.
 - 3.5. Update the probabilistic classifier: $\mathbf{h} \leftarrow \mathbf{h} \cup \{\mathbf{h}^i\}$, $\alpha = \alpha \cup \{\alpha^i\}$,
 $\mathbf{a}^j \leftarrow \mathbf{a}^j \times \alpha^i$ for $j \in [i-1]$.

Set of all feasible \mathbf{C}

Utility function Ψ

The Frank Wolfe **guarantees** to find the optimal classifier if Ψ is **concave** over $\mathcal{C}_P^{\otimes k}$, **L-Lipschitz**, and **β -smooth** w.r.t. the 1-norm.

Read the paper

Try our library

<https://github.com/mwydmuch/xCOLUMNS>

Experimental results

We compare classifiers found using the Frank Wolfe algorithm with popular re-weighting strategies:

Inference strategy	Mediamill						RCV1X					
	Instance @5		Macro @5		Instance @5		Macro @5		Precision		Recall	
TOP-K	51.96	62.04	12.85	8.75	7.71	52.30	81.96	12.77	7.61	7.64		
TOP-K+ w^{POW}	47.68	56.62	13.00	17.37	12.64	48.48	77.18	14.69	17.66	13.64		
TOP-K+ w^{LOG}	35.40	39.96	11.38	15.33	10.95	51.18	80.49	16.03	11.75	11.29		

Inference strategy	Mediamill						RCV1X					
	Instance @5		Macro @5		Instance @5		Macro @5		Precision		Recall	
MACRO-P _{FW}	6.99	8.96	17.29	8.79	3.17	29.40	49.81	21.69	5.72	5.31		
MACRO-R _{PRIOR}	7.38	7.25	8.91	26.50	6.71	34.77	56.28	13.13	24.59	12.77		
MACRO-R _{FW}	7.38	7.25	8.91	26.50	6.71	34.15	55.24	13.15	24.89	12.73		
MACRO-F1 _{FW}	43.57	51.60	15.20	15.05	13.82	44.42	71.86	21.96	12.25	13.68		

Interpolated metrics

– Optimizing macro-measures incurs a **significant drop** in instance-wise measures.
 – To achieve the desired trade-off between tail and head label performance, a straight-forward **combination** between standard instance-wise precision@ k , and a macro-average @ k metric can be used, e.g.:

$$\Psi(\mathbf{C}) = (1 - \lambda)\Psi_{\text{Instance-Precision}@k}(\mathbf{C}) + \lambda\Psi_{\text{Macro-F1}@k}(\mathbf{C})$$

MEDIAMILL RCV1X

A combination of a macro-metric and instance precision-at- k can achieve good results on both metrics **simultaneously**!