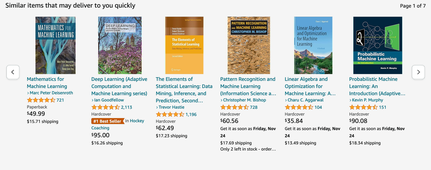
Consistent algorithms for multi-label classification with macro-at-k metrics

Erik Schultheis[1] Wojtek Kotłowski[2] Marek Wydmuch[2] Rohit Babbar[1,3] Strom Borman[4] Krzysztof Dembczyński[2,5]

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

Multi-label classification with budget at k

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

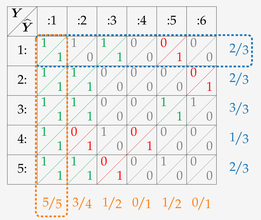


Macro-at-k matrics

Instead of calculating and averaging performance instance-wise, calculate per-label ψ^j and average over labels (label-wise):

$$\Psi@k(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) = \sum_{j=1}^{m} \psi^j(\boldsymbol{y}_{:j}, \widehat{\boldsymbol{y}}_{:j}) .$$

- Can balance contribution from all labels.
- **Reduces to macro-average** if all ψ^j are the same.
- Covers (instance) Precision@k and Hamming score.



Precision@3: instance ≈ 66%, macro ≈ 46%

Metric	$\psi(\mathrm{tp}, \mathrm{fp}, \mathrm{fn}, \mathrm{tn})$
Precision	$\frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fp}}$
Recall	$\frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fn}}$
F_1	$\frac{2\mathrm{tp}}{2\mathrm{tp}+\mathrm{fn}+\mathrm{fp}}$
Jaccard	$\frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fp}+\mathrm{fn}}$
Ballanced Acc., G-Mean, etc.	

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

$$\widehat{\mathbf{C}}(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) := [\boldsymbol{C}(\boldsymbol{y}_{:1}, \widehat{\boldsymbol{y}}_{:1}), \ldots, \boldsymbol{C}(\boldsymbol{y}_{:m}, \widehat{\boldsymbol{y}}_{:m})] .$$

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

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

We call metrics Ψ defined on $\widehat{\boldsymbol{C}}$ **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:

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

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:

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

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

The optimal PU classifier

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

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

- Set of confusion tensors achievable by $\boldsymbol{h}$: $\mathcal{C}_{\mathrm{P}}^{@k} := \{\mathbf{C}(\boldsymbol{h}) : \boldsymbol{h} \in \mathcal{H}\}$.
- We proof that $\mathcal{C}_{\mathrm{P}}^{@k}$ is compact, simplifying the subsequent analysis.

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

$$\boldsymbol{h}^\star(\boldsymbol{x}) = \mathrm{top}_k(\boldsymbol{a} \odot \boldsymbol{\eta}(\boldsymbol{x}) + \boldsymbol{b}),$$

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

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

- The optimal classifier for **general confusion tensor metrics:**
  - *If:* $\boldsymbol{\eta}(\boldsymbol{x})$ *is* ***absolutely continuous***,
  - *and* Ψ *is* ***strictly monotonic*** *and* ***differentiable***,

  there exists an optimal $\mathbf{C}^\star \in \mathcal{C}_{\mathrm{P}}^{@k}$*, that is* $\Psi(\mathbf{C}^\star) = \Psi^\star$*. Any classifier* $\boldsymbol{h}^\star$ *maximizing the linear utility* $\mathbf{G} \cdot \mathbf{C}(\boldsymbol{h})$ *with* $\mathbf{G} = \nabla_{\mathbf{C}}\Psi(\boldsymbol{C}^\star)$ *also maximizes* $\Psi(\boldsymbol{h})$.

Difficulty of optimizing metrics-at-k

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

Distribution $\boldsymbol{A}$ | $\boldsymbol{B}$

	P	η_1	η_2	η_3
$\boldsymbol{x}_1$	0.5	0.4	0.2	0.6
$\boldsymbol{x}_2$	0.5	0.8	0.4	**0.4** | **0.8**

$\Psi_{\text{Jaccard}}(\mathbf{C}(\boldsymbol{h}^\star_A, A)) \approx 0.454,$

Optimal $\boldsymbol{h}^\star_A(\boldsymbol{x})$ | $\boldsymbol{h}^\star_B(\boldsymbol{x})$

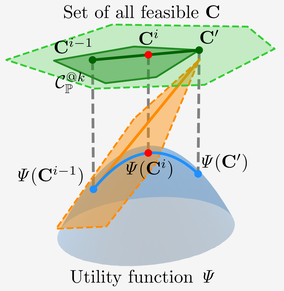
	π_1	π_2	π_3
$\boldsymbol{x}_1$	**1** | **0**	**0** | **1**	1
$\boldsymbol{x}_2$	1	1	0

$\Psi_{\text{Jaccard}}(\mathbf{C}(\boldsymbol{h}^\star_B, B)) \approx 0.471.$

Finding the optimal PU classifier

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

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



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

Read the paper

Try our library

Experimental results

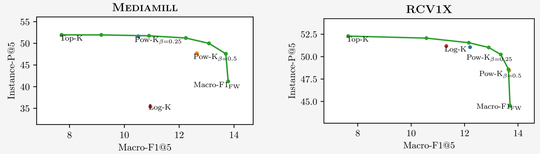
We compare classifiers found using the Frank Wolfe algorithm with with popular re-weighting strategies:

Inference strategy	Mediamill Instance @5 Prec	Mediamill Instance @5 Rec	Mediamill Macro @5 Prec	Mediamill Macro @5 Rec	Mediamill Macro @5 F1	RCV1X Instance @5 Prec	RCV1X Instance @5 Rec	RCV1X Macro @5 Prec	RCV1X Macro @5 Rec	RCV1X Macro @5 F1
Top-K	**51.96**	**62.04**	12.85	8.75	7.71	**52.30**	**81.96**	12.77	7.61	7.64
Top-K+$\boldsymbol{w}^{\mathrm{POW}}$	47.68	56.62	13.00	17.37	12.64	48.48	77.18	14.69	17.66	13.64
Top-K+$\boldsymbol{w}^{\mathrm{LOG}}$	35.40	39.96	11.38	15.33	10.95	51.18	80.49	16.03	11.75	11.29
Macro-P_{FW}	6.99	8.96	**17.29**	8.79	3.17	29.40	49.81	**21.69**	5.72	5.31
Macro-$\mathrm{R}_{\mathrm{PRIOR}}$	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-$\mathrm{F1}_{\mathrm{FW}}$	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.:

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



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

ICLR '24, May 7–11, 2024, Vienna, Austria