Efficient Learning of Mahalanobis Metrics for Ranking

Daryl Lim and Gert Lanckriet

[1]Dept. of Electrical and Computer Engineering
University of California, San Diego

June 22, 2014

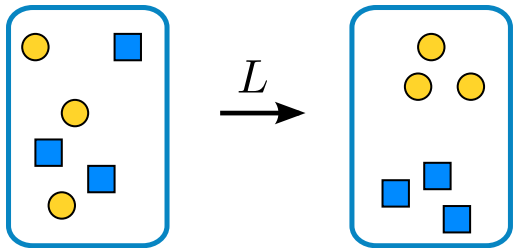
Metric Learning Algorithms

- Metric learning algorithms learn a good linear transformation L, or equivalently a metric $W = L^{\mathsf{T}} L$ from data.

- Given two points x,y, squared distance between them after transformation is

$$\|Lx - Ly\|^2 = (x - y)^{\mathsf{T}} W (x - y)$$

- Similar points should be close under the metric, and dissimilar points far apart

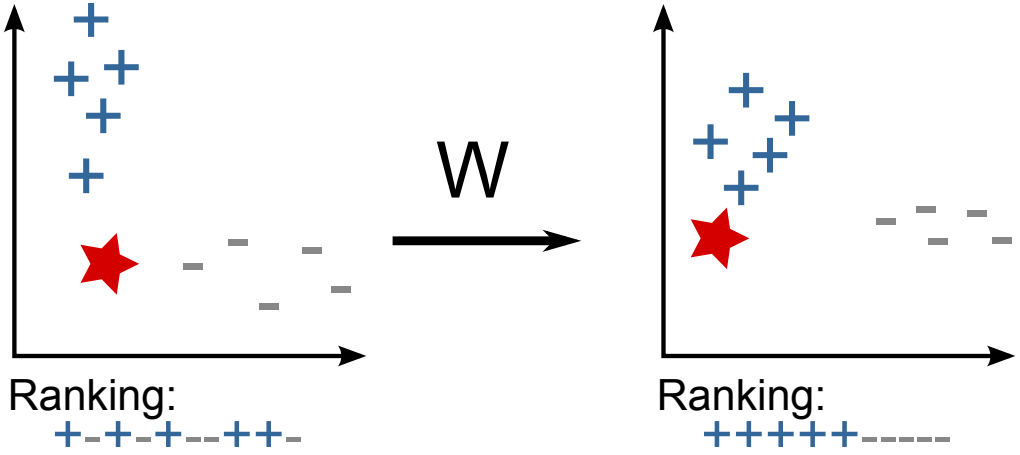


Metric Learning as Ranking

- We have training data $\mathcal{X} \in \mathbb{R}^{d \times n}$
- For every query $q \in \mathcal{X}$, we are given $\mathcal{X}_q^+, \mathcal{X}_q^-$ subsets of training set that are similar/dissimilar (binary relevance)
  - e.g. derived from class labels (transitive), NN in graphs (nontransitive), etc
- Want to learn W such that on average, points in $\mathcal{X}_q^+$ are close to q
- Don't care about absolute distance, just about relative order.

Metric Learning as Ranking

- Given a query point, ordering the training set by distance from query point should give a "good" ranking



Metric Learning as Ranking

Desiderata:

- Given a query, similar points should appear at the top of the ranking induced by W
  - Optimize a ranking loss that focuses on top of the ranking
- Our approach should scale to large training sets
  - Derive an efficient stochastic gradient descent algorithm
- Our approach should scale to high-dimensional data
  - Optimize with respect to a low-rank metric using Riemannian optimization

General Approach

Our optimization can be written as

$$\min_{\substack{W \succeq 0 \\ \text{rank}(W)=m}} \sum_{q \in \mathcal{X}} \text{Loss}(q, W) + \Omega(W)$$

and typically $m << d$, the data dimension

What loss to choose?

Ranking Metrics

- For a given query q and $x^+ \in \mathcal{X}_q^+$ define

$$\text{rank}(x^+) = \sum_{x^- \in \mathcal{X}_q^-} I(d_W(q, x^-) < d_W(q, x^+))$$

- $\text{rank}(x^+)$ is the number of $x^- \in \mathcal{X}_q^-$ which lie between q and x^+

Ranking Metrics

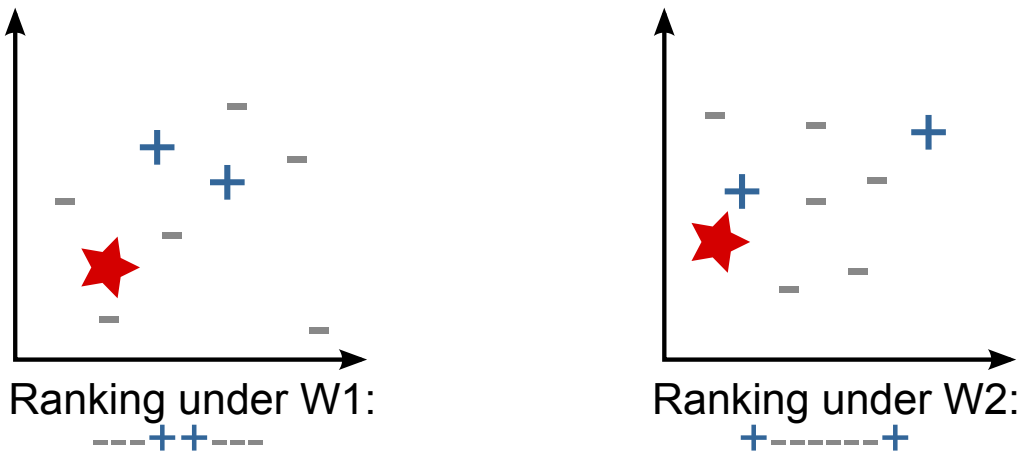
- Area under ROC curve (AUC) is a common measure of ranking quality

$$\begin{aligned}\text{AUC} &= \frac{1}{|\mathcal{X}_q^+||\mathcal{X}_q^-|} \sum_{x^+ \in \mathcal{X}_q^+} \sum_{x^- \in \mathcal{X}_q^-} I(d_W(q, x^-) < d_W(q, x^+)) \\ &= \frac{1}{|\mathcal{X}_q^+||\mathcal{X}_q^-|} \sum_{x^+ \in \mathcal{X}_q^+} \text{rank}(x^+)\end{aligned}$$

- Minimizes mean rank of positives

Ranking Metrics

Consider two metrics W1, W2 which induce the following embeddings:



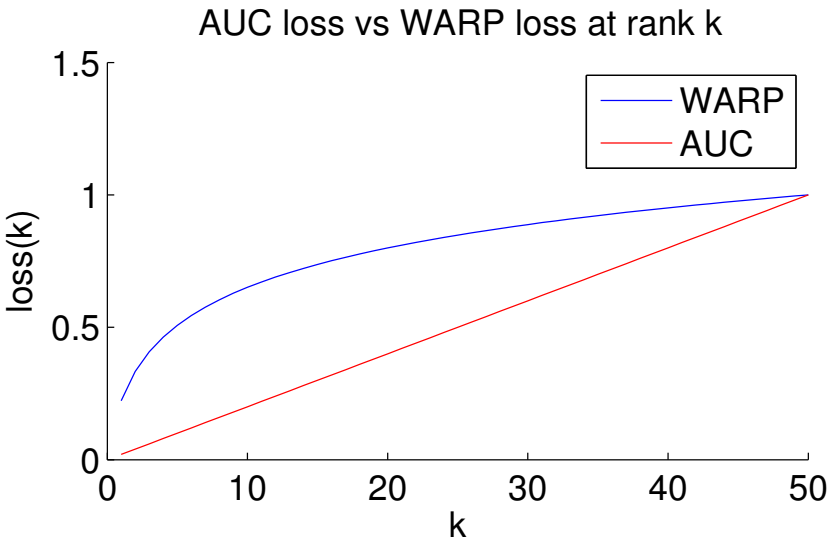
- AUC of both metrics is the same!
- Top-of-the-ranking performance is preferred

WARP Loss

- WARP loss (Weston et al., 2010)

$$\frac{1}{Z}\sum_{x^+\in\mathcal{X}^+}\mathcal{L}(\text{rank}(x^+)),\quad \mathcal{L}(k)=\sum_{i=1}^{k}\alpha_i,\quad \alpha_1\geq\cdots\alpha_k\geq 0$$

- choice of $\alpha_i=\frac{1}{i}$ promotes ranking of positives at the top.



WARP Loss

How to optimize for WARP loss which is discontinuous?

- Define $\text{viol}(x^+) = \{x^- \in \mathcal{X}_q^- : d_W(q, x^-) < d_W(q, x^+) + 1\}$
- Define $\text{rankv}(x^+) = |viol(x^+)|$

- Can make WARP loss continuous using the upper bound

$$\begin{aligned} &\mathcal{L}(\text{rank}(x^+)) \\ \leq &\mathcal{L}(\text{rankv}(x^+)) \sum_{x^- \in \text{viol}(x^+)} \frac{|1 - d_W(q, x^+) + d_W(q, x^-)|_+}{\text{rankv}(x^+)} \end{aligned}$$

WARP Loss

Can now minimize via stochastic gradient descent.

$$\min \; E_{(q,x^+,x^-)}\left[\mathcal{L}(\text{rankv}(x^+))\,|1-d_W(q,x^+)+d_W(q,x^-)|_+ + \Omega(W)\right]$$

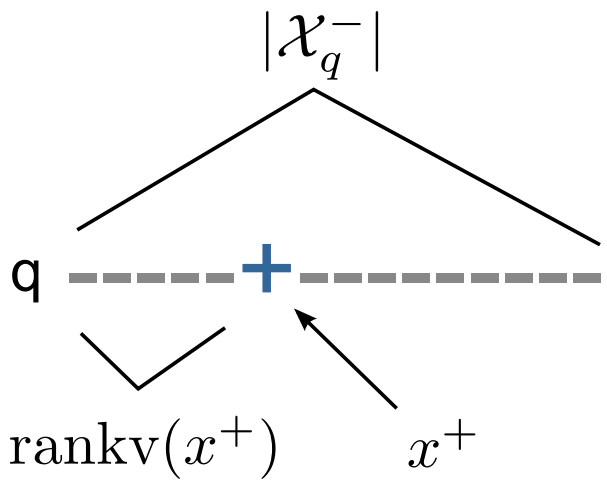
- Problem: $\text{rankv}(x^+)$ is expensive to compute

  Have to calculate $d_W(q,x^-)$ $|\mathcal{X}_q^-|$ times, and usually $|\mathcal{X}_q^-| \approx$ #training points.

- Solution: Approximate it

Approximating rankv(x^+)

Sample x^- from $\mathcal{X}_q^-$ until $d_W(q, x^-) < d_W(q, x^+) + 1$

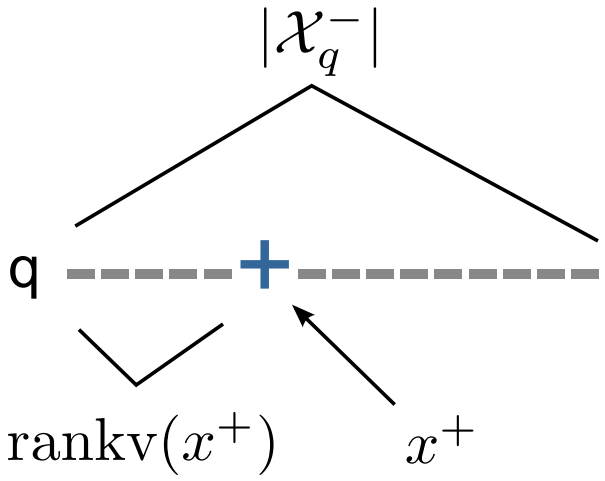


Number of trials, N_k, is a geometric random variable with parameter $p = \frac{rankv(x^+)}{|\mathcal{X}_q^-|}$

Since $E(N_k) = \frac{1}{p}$, approximate rankv(x^+) by $\left\lfloor \frac{|\mathcal{X}_q^-|}{N_k} \right\rfloor$

Approximating rankv(x^+)

Sample x^- from $\mathcal{X}_q^-$ until $d_W(q, x^-) < d_W(q, x^+) + 1$



Equivalent to optimizing

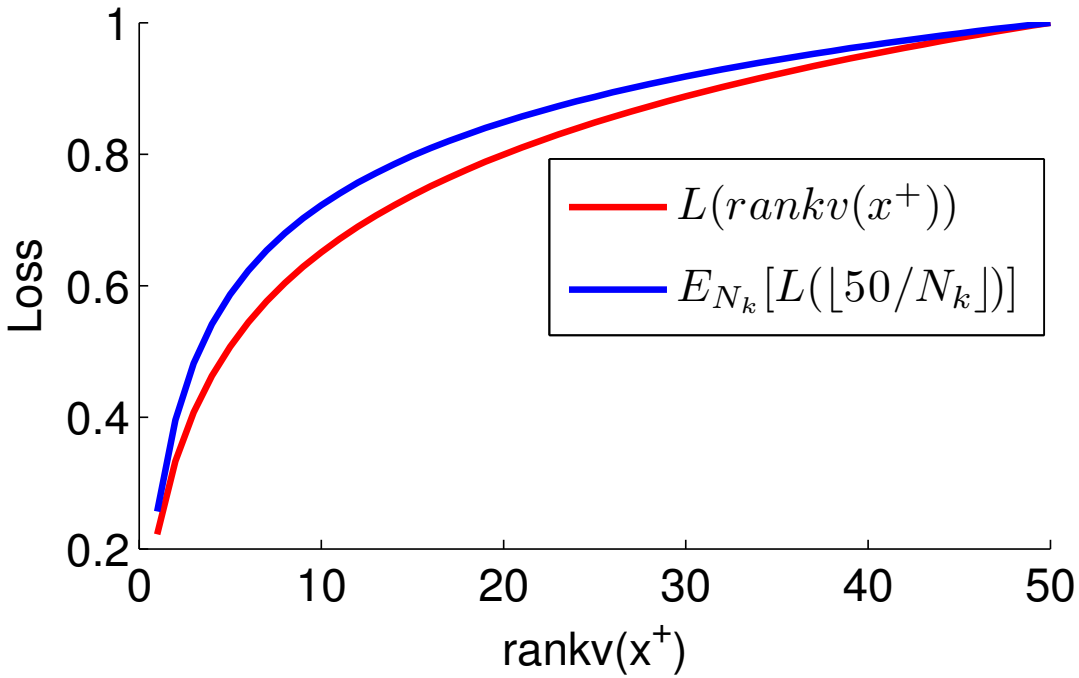
$$\mathbb{E}\left[\ \mathbb{E}_{N_k \sim G_k}\left[\mathcal{L}\left(\left\lfloor \frac{|\mathcal{X}_q^-|}{N_k} \right\rfloor\right)\right][1 - f_q(x^+) + f_q(x^-)]_+ + \Omega(W)\right]$$

instead of

$$\mathbb{E}\left[\ \mathcal{L}(\text{rankv}(x^+))[1 - f_q(x^+) + f_q(x^-)]_+ + \Omega(W)\right]$$

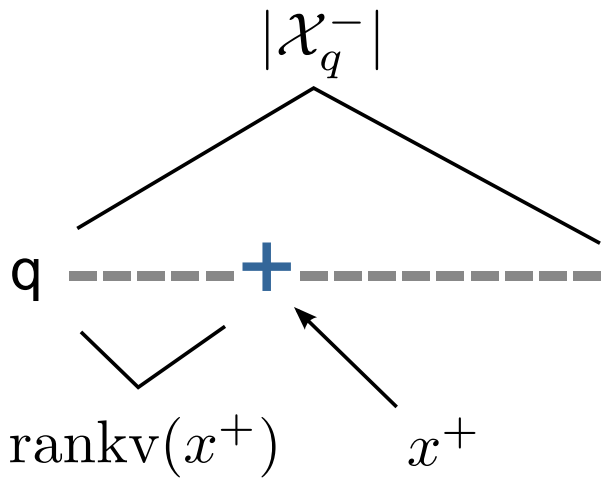
Approximating rankv(x^+)

$$\mathbb{E}_{N_k \sim G_k}\left[\mathcal{L}\left(\left\lfloor \frac{|\mathcal{X}_q^-|}{N_k} \right\rfloor\right)\right] \text{ vs } \mathcal{L}(\text{rankv}(x^+))$$



Approximating rankv(x^+)

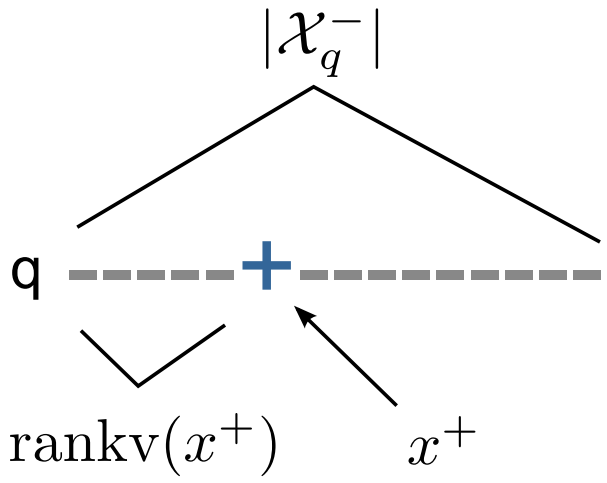
Sample x^- from $\mathcal{X}_q^-$ until $d_W(q, x^-) < d_W(q, x^+) + 1$



In the worst case, stop sampling once $N_k > |X_q^-|$, since $\mathcal{L}(\lfloor \frac{|\mathcal{X}_q^-|}{N_k} \rfloor) = 0$, so loss component of the gradient is zero.

If $rankv(x^+)$ is small, obtaining N_k is still expensive as $|\mathcal{X}_q^-|$ is on the same order as training set size.

Approximating rankv(x^+)



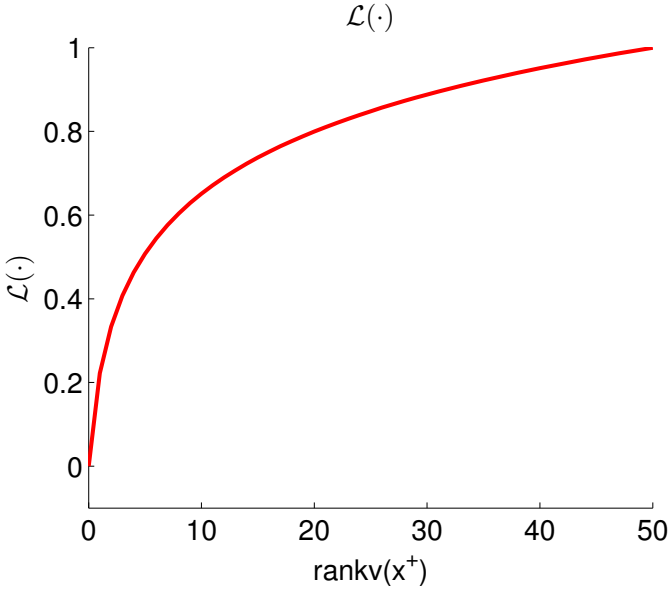
Proposal: Stop sampling once $N_k > \frac{|X_q^-|}{\gamma}$, where $\gamma > 0$ is a hyperparameter.

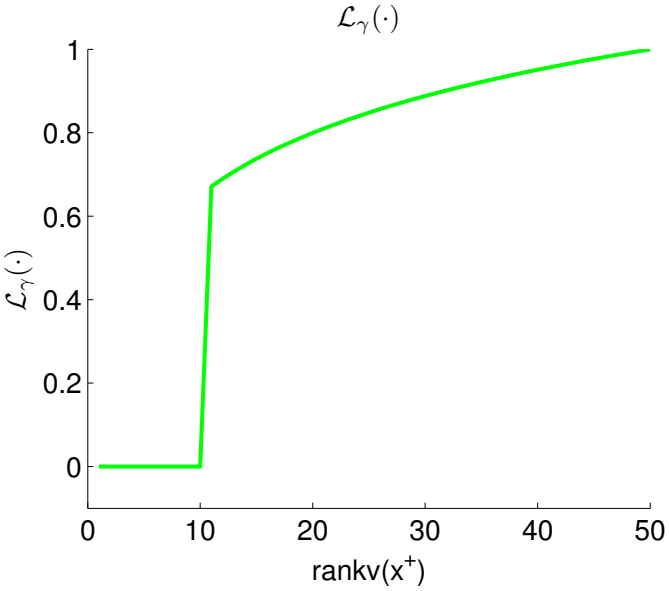
- Can't take a gradient step on loss (since we haven't found a violator) so just move on to the next point.
- This is equivalent to assuming that $\left\lfloor \frac{|\mathcal{X}_q^-|}{N_k} \right\rfloor = 0$ when it may not be.

Approximating rankv(x^+)

Proposed scheme is exactly equivalent to using a modified loss function $\mathcal{L}_\gamma(\cdot)$ in place of $\mathcal{L}(\cdot)$:

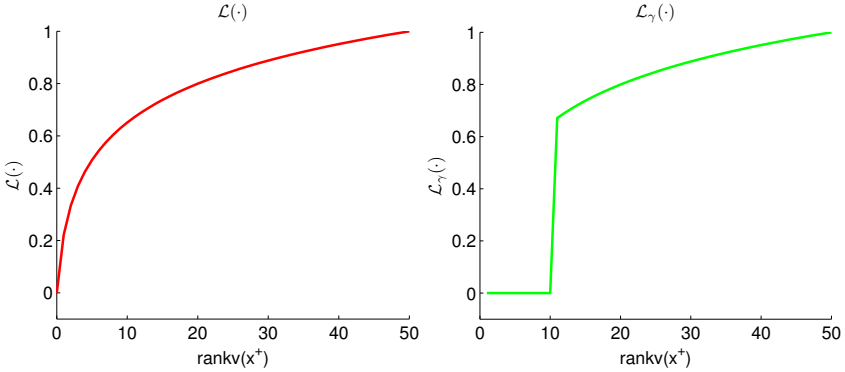
$$\mathcal{L}_\gamma(k) = \begin{cases} \mathcal{L}(k) & k \geq \gamma \\ 0 & k < \gamma \end{cases}$$





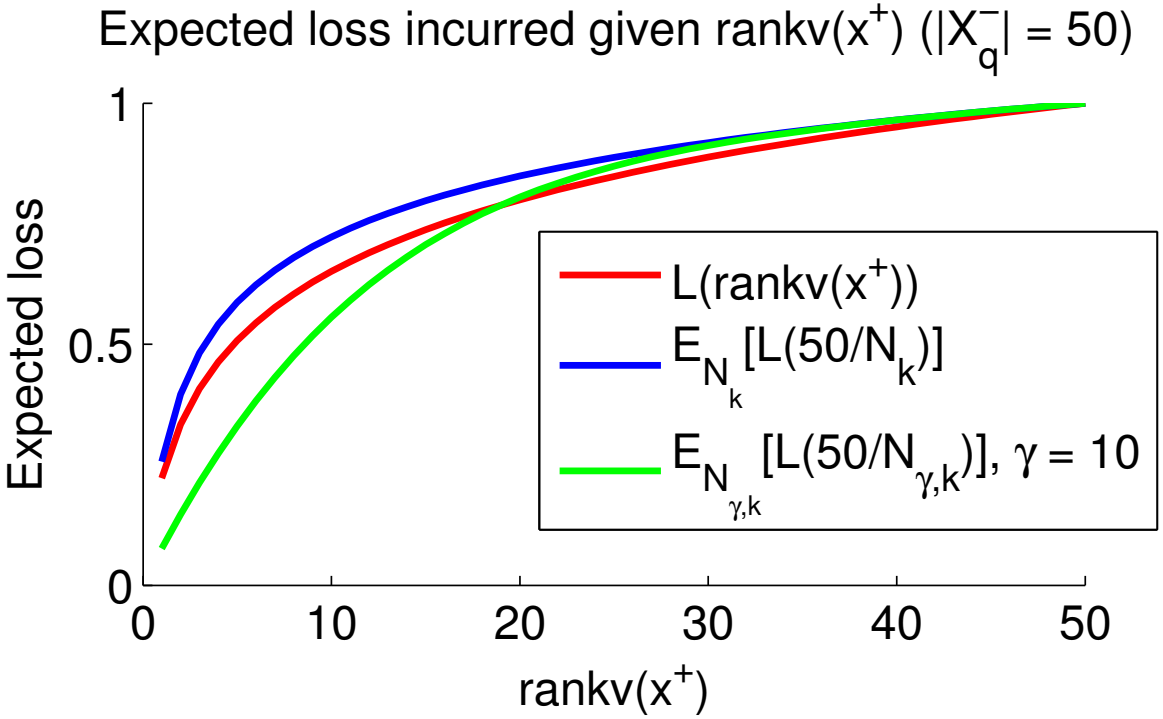
New loss function is nonconvex and assigns zero loss for rankv(x^+) $< \gamma$. Doesn't seem like it should work!

Approximating rankv(x^+)



- New loss function is nonconvex and assigns zero loss for rankv(x^+) $< \gamma$. Doesn't seem like it should work!
- However recall we are optimizing $\mathbb{E}_{N_k \sim G_k}\left[\mathcal{L}_\gamma\left(\left\lfloor \frac{|\mathcal{X}_q^-|}{N_k} \right\rfloor\right)\right]$ instead of $\mathcal{L}_\gamma(k)$
- This expected loss turns out to be qualitatively similar to the WARP loss.

Approximating rankv(x^+)



In fact, as long as $\mathcal{L}$ convex nondecreasing, $\mathbb{E}_{N_k}(\mathcal{L}_\gamma)$ is also convex nondecreasing for ANY γ (journal version)

In the extreme case where we set $\gamma > |\mathcal{X}_q^-|$, reduces to AUC

Optimization

Optimize via stochastic gradient descent:

1. Draw random q
2. Draw x^+ from X_q^+
3. Sample x^- from $|\mathcal{X}_q^-|$ until violator found or $N_k > |X_q^-|/\gamma$;
4. Calculate $\nabla_W \mathcal{L}(\lfloor |\mathcal{X}_q^-|/N_k \rfloor)]|1 - d_W(q, x^+) + d_W(q, x^-)|_+$, perform gradient descent on W

How to ensure that W satisfies $\text{rank}(W) = m, W \succeq 0$?

Optimization

How to ensure that W satisfies $\text{rank}(W) = m$, $W \succeq 0$?

Two common methods:

- Projected gradient descent:
  - After gradient step, factor $W = Q\Lambda Q^{\mathsf{T}}$, and threshold/retain m largest eigenvalues of Λ.
  - Problem: $O(d^3)$!
- Optimize Factored form
  - Let $W = LL^{\mathsf{T}}$, where $L \in \mathbb{R}^{d \times m}$. Rewrite optimization in terms of L instead of W.
  - Problem: No isolated minima, very sensitive to initialization/step size, poor convergence/performance.
  - if $LL^{\mathsf{T}} = W$, then $MM^{\mathsf{T}} = W$ where $M = GL$, G is an orthonormal perturbation of identity

Riemannian Optimization

Solution: Riemannian Optimization!

- PSD matrices of fixed rank form an embedded Riemannian manifold.
- We can optimize a function directly over the set of rank-m, $d \times d$ PSD matrices
- Gradient descent requires 3 steps:
  1. $W^+ = W - \eta \nabla_W f(W)$
  2. Project W^+ onto tangent space at W
  3. Retract W^+ onto manifold
- Seems complicated, but is relatively cheap to compute. $O(d \max(m^2, r^2))$ where r is rank of $\nabla_W f(W)$

Riemannian Optimization

Algorithm 1 Symmetric_gradient_update

Input: Initial matrix $L \in \mathbb{R}^{d \times m}$ such that $W = LL^{\mathsf{T}}$, U, V such that $-\eta \nabla_W = UV^{\mathsf{T}}$

Output: M such that $MM^{\mathsf{T}} = \mathcal{R}_W(P_{T_W}(W - \eta \nabla W))$

1: $L^{\dagger} = (L^{\mathsf{T}} L)^{-1} L^{\mathsf{T}}$

2: $A_1 = L^{\dagger} U;\ A_2 = L^{\dagger} V;\ S = A_1^{\mathsf{T}} A_2;\ \hat{A}_1 = L A_1$

3: **return** $M = L + (U - \frac{1}{2}\hat{A}_1 + (\frac{3}{8}\hat{A}_1 - \frac{1}{2}U)S)A_2^{\mathsf{T}}$

Riemannian Optimization

Advantages of Riemannian Optimization

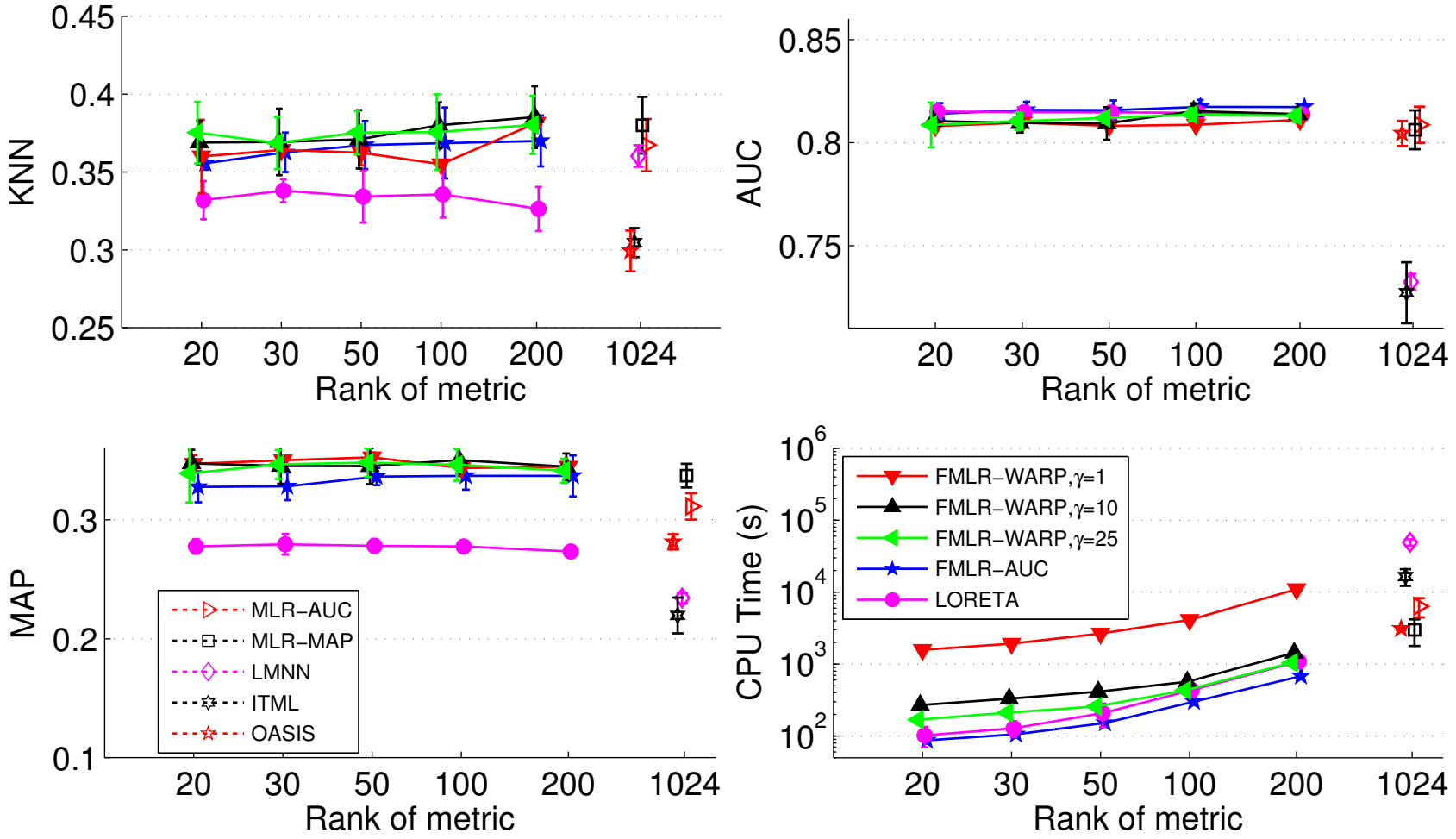
- SGD on embedded Riemannian manifold converges to local minimum of the loss
- Only need to store factored form (L such that $\mathrm{LL}^{\mathsf{T}} = W$) in memory
- Fewer local minima + isolated minima
- Much more robust to step size/initialization

Experiments

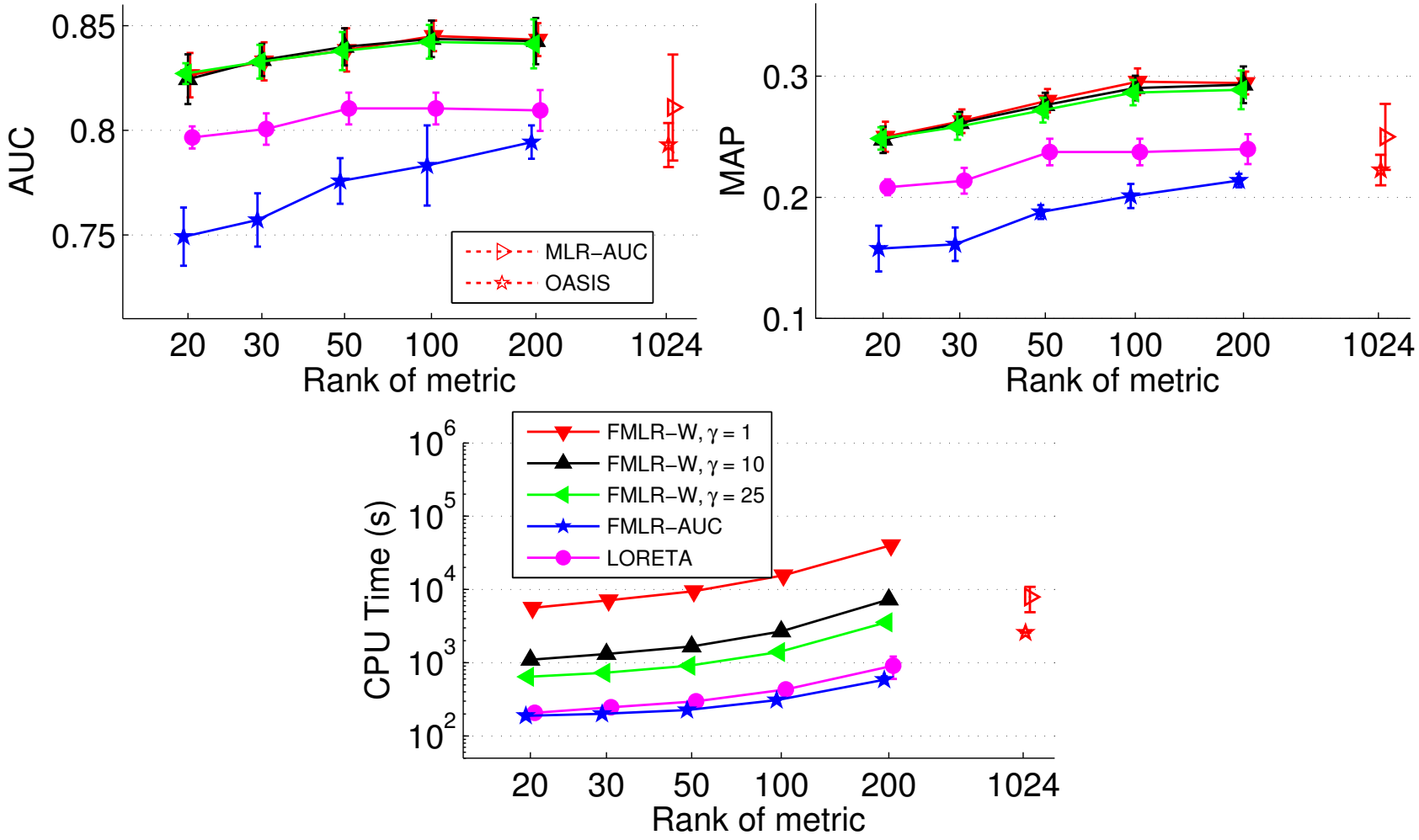
Datasets:

- ImageNet
  - d = 1024, n = 1000
  - Images encoded with 1024-dimensional histograms
  - Similarity defined classwise (20 classes)
- CAL10K
  - d = 2048, n = 2129
  - Songs encoded with 2048-dimensional histogram of VQ representations
  - Similarity defined by collaborative filter data
- MagnaTagaTune
  - d = 1024, n = 8577
  - Songs encoded with 1024-dimensional representation (sparse coding)
  - Similarity defined by tag vector correlations

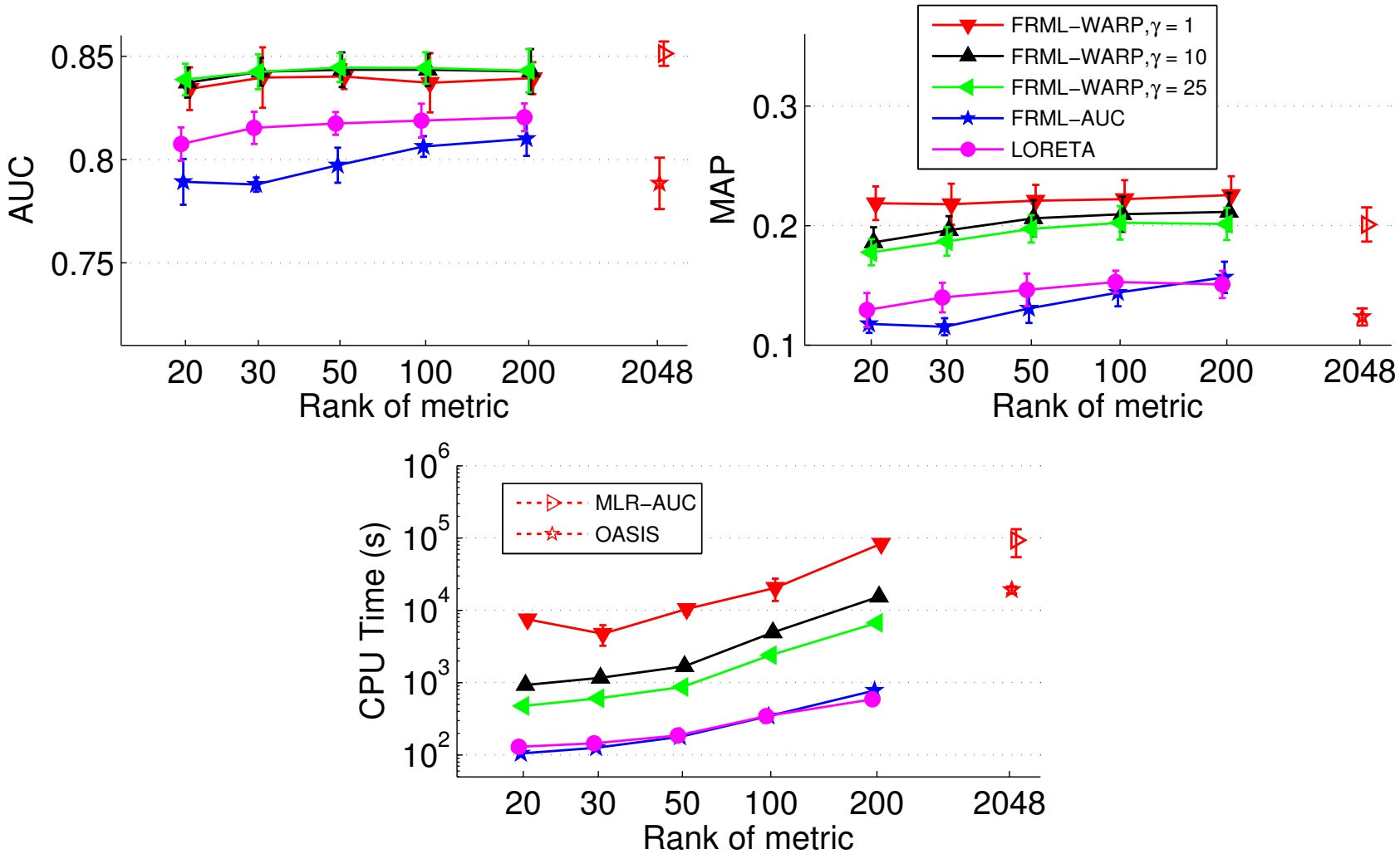
Experiments: ImageNet



Experiments: Magnatagatune



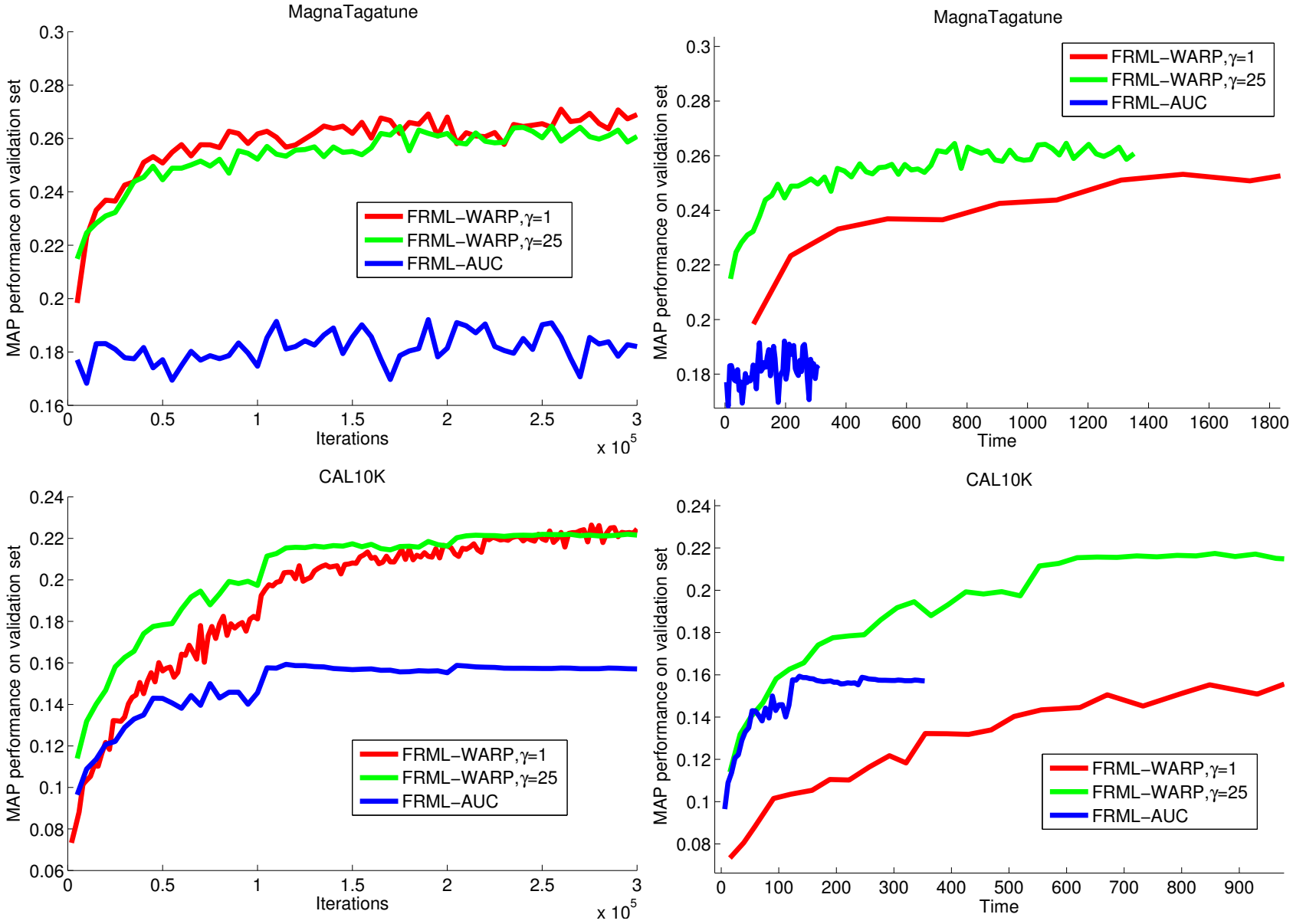
Experiments: CAL10K



Experiments: Precision at K

		FRML-AUC	FRML-WARP			LORETA	OASIS	MLR-AUC	MLR-MAP	LMNN	ITML
			γ=1	γ=10	γ=25						
ImageNet	P@1	0.366	0.377	**0.380**	0.378	0.339	0.271	0.354	0.375	0.355	0.292
	P@10	0.344	0.357	**0.359**	0.356	0.324	0.280	0.330	0.357	0.320	0.264
CAL10K	P@1	0.262	**0.328**	0.297	0.292	0.221	0.182	0.271			
	P@10	0.226	**0.294**	0.262	0.256	0.193	0.160	0.241			
MagnaTagatune	P@1	0.470	0.521	**0.524**	0.519	0.444	0.460	0.467			
	P@10	0.414	0.485	**0.489**	0.480	0.412	0.408	0.425			

Experiments: Convergence



Experiments: Covertype

	AUC	MAP	P@1	P@10
FRML-WARP	0.843	**0.511**	**0.811**	**0.689**
FRML-AUC	**0.852**	0.477	0.749	0.648
LORETA	0.823	0.427	0.804	0.671
MLR-AUC	0.851	0.481	0.762	0.665
OASIS	0.792	0.417	0.792	0.674
LMNN	0.661	0.307	0.782	0.666
ITML	0.835	0.458	0.762	0.663

Conclusion

- Developed an efficient algorithm to optimize top-of-the ranking performance for metric learning
- Riemannian optimization for PSD is stable and fast!
- Given a fixed training time budget, FRML-WARP with gamma set appropriately gets the best performance

Conclusion

Thank you!