Learning to Rank and Quadratic Assignment

Thomas Mensink[*]
TVPA - XRCE &
LEAR - INRIA
Grenoble, France

Jakob Verbeek
LEAR Team
INRIA Rhône-Alpes
Grenoble, France

Tiberio Caetano
Machine Learning Group
NICTA
Sydney, Australia

Abstract

In this paper we show that the optimization of several ranking-based performance measures, such as precision-at-k and average-precision, is intimately related to the solution of quadratic assignment problems. Both the task of test-time prediction of the best ranking and the task of constraint generation in estimators based on structured support vector machines can all be seen as special cases of quadratic assignment problems. Although such problems are in general NP-hard, we identify a polynomially-solvable subclass (for both inference and learning) that still enables the modeling of a substantial number of pairwise rank interactions. We show preliminary results on a public benchmark image annotation data set, which indicates that this model can deliver higher performance over ranking models without pairwise rank dependencies.

1 Introduction

In document retrieval, image retrieval, and image annotation problems, performance is often evaluated based on a ranking of the items, *e.g.* the ranking of documents for a given query, or the ranking of annotation terms for a given image. Well known ranking evaluation measures include *precision-at-k* (p@k), and *average precision* (AP). For ranking, the prediction problem does not decompose into a set of independent predictions, where a separate loss is incurred for each prediction, which is for example the case for classification problems. This multi-variate nature of the loss function has hindered the development of methods that directly optimize for ranking losses. Instead, the parameters are often trained using a proxy loss. For example, in image annotation it is common practice to learn a classifier per annotation label, and the labels are then ranked by the classification score. Following the development of the structured SVM framework [1, 2], learning ranking functions has become an active area of research, see *e.g.* [3, 4, 5, 6, 7]. Once a model has been learned, these methods compute a ranking by sorting items based on a score computed for each item. Methods to learn the score functions optimized for a variety of different performance measures have been developed, including AP in [3], and p@k in [4].

A feature that is missing in the existing methods, however, is the ability to encode pairwise correlations between items, *i.e.* terms in the score function that will encourage some items to be ranked at nearby locations, or conversely to be ranked far apart. In ranking problems such as image annotation we expect strong correlations between the relevance of labels. Some labels will be positively correlated, such as *beach* and *sand*, while others will be negatively correlated such as *sofa* and *car*. Indeed, models for image understanding that exploit such correlations have been reported to outperform models that do not, see *e.g.* [8, 9]. However these models are trained using maximum likelihood criteria, and it is not obvious how they can be trained for ranking based losses. Recently a solution was given for the problem of optimizing structured losses for *multi-label* classification under pairwise label interactions for any graph topology [10], but multi-label losses are easier to deal with than permutation-based losses since optimization over the permutation group is typically

[*] This research was performed during a research visit of the SML Group of NICTA - Canberra, Australia.

harder than over the power set. A principled strategy for optimising ranking losses under pairwise item interactions is still an open problem.

In this paper we consider learning structured score functions, where items have interdependencies, while optimizing directly for ranking losses. Our contributions are the following: (i) We define a generic family of ranking models where inference takes the form of a quadratic assignment problem (QAP), which is generally NP-hard. (ii) We identify a subclass of such models with linear loss functions and star-structured interactions, where inference can be performed in polynomial time. (iii) In particular, for the precision-at-k loss, this leads to a practical model where inference is linear in the number of elements to rank, and exponential in the size of the number of elements in the fully connected clique at the center of the star.

In the next section we review the most relevant related work. In Section 3 we introduce a general framework to learn ranking models, which includes earlier approaches such as [3, 4]. In Section 4 we consider the case of learning structured models for the p@k loss. In Section 5 we apply the model to an image annotation problem, and we summarize our conclusions in Section 6.

2 Related work

We use structured SVMs [1, 2] to learn ranking functions, as in [3, 4, 5]. In the latter works a ranking is obtained by sorting items according to associated scores which are computed independently for each item. For example in document retrieval, the score s_i for document $\boldsymbol{x}_i$ given a query q is computed as $s_i = \langle \phi(\boldsymbol{x}_i, q), \boldsymbol{w} \rangle$, where a single $\boldsymbol{w}$ is learned. Similarly in image annotation the score s_i for a label i given an image $\boldsymbol{x}$ is computed as $s_i = \langle \phi(\boldsymbol{x}), \boldsymbol{w}_i \rangle$, where the features are identical, but a different weight vector $\boldsymbol{w}_i$ is learned for each label. In either case, for each item to be ranked a linear score is computed, that does not depend on the score of other items.

The class of linear ranking losses is defined in [4] as the set of loss functions that can be written as a linear function of the permutation matrix Π that encodes the ranking: $\Delta(\Pi, \boldsymbol{z}) = \langle \Pi \boldsymbol{a}, \boldsymbol{b}(\boldsymbol{z}) \rangle$, where $\boldsymbol{a}$ is a loss-specific vector, and $\boldsymbol{b}(\boldsymbol{z})$ is a vector which encodes ground truth relevance annotation ($z_i = 1$ for relevant documents, and $z_i = 0$ for irrelevant documents). The class of linear loss functions includes the winner-takes-all loss (*i.e.* measuring wether the first item is correct), mean reciprocal rank (*i.e.* defined as the inverse of the rank of the relevant item, assuming there is only one), and p@k (*i.e.* how many items are there in the top-k) performance measures. For example, the p@k loss is obtained by setting $\boldsymbol{b} = \boldsymbol{z}$ and $a_i = 1 - \frac{1}{k}$ for $1 \leq i \leq k$, in this manner placing an irrelevant document item in the top k reduces the p@k by $\frac{1}{k}$, and increases the loss by the same amount. In general, optimizing for a linear loss using independent scoring of items, leads to a linear assignment problem [4], which has a worst case complexity that is cubic in the number of elements to rank. An efficient approximate online learning approach for image annotation using a linear loss was introduced in [5], for cases where it is too expensive to compute the score for all elements.

Average-precision (AP) provides a single-figure measure of quality across recall levels by averaging p@k over all values of k occupied by relevant documents. The AP measure, however, is not linear in the permutation matrix, but quadratic since the contribution of item i to the performance depends also on the relevance of the items ranked before and after item i. To see this, let A be a lower triangular matrix with $\frac{1}{k}$ on the first k entries of the k-th row. Then $[\boldsymbol{A}\Pi\boldsymbol{z}]_k$ will contain the p@k for each choice of k. The AP is obtained by averaging the p@k values over all positions of relevant documents, thus by $\langle \Pi\boldsymbol{z}, \boldsymbol{A}\Pi\boldsymbol{z} \rangle / (\mathbf{1}^\top \boldsymbol{z}) = \text{Tr}\{\boldsymbol{A}\Pi\boldsymbol{B}\Pi^\top\}$, where $\boldsymbol{B} = \boldsymbol{z}\boldsymbol{z}^\top / (\mathbf{1}^\top \boldsymbol{z})$. In general, for such quadratic loss functions, one has to solve NP-hard quadratic assignment problems during learning. For the special case of the AP loss, however, a model where learning scales only quadratically with the number of items to rank has been found [3]. In the next section we consider similar quadratic forms for define *score functions* that will encourage pairs or items to be ranked close together or far apart.

3 Learning quadratic ranking functions

When learning ranking models the training set consists of queries/images with corresponding relevance labeling of the items to be ranked. We use $\boldsymbol{x}$ to denote the input data, which can be either features for documents to be ranked for a query, or features of an image for which we want to rank labels. We will use Π to denote permutation matrices that encode the ranking of the items. Our goal is to learn a function that will compute from an input $\boldsymbol{x}$ a ranking Π that will incur a small loss $\Delta(\Pi, \boldsymbol{z})$, where $\boldsymbol{z}$ denotes the ground-truth relevance labeling of the item.

Following the structured SVM learning paradigm [1, 2] we define a score function $f(\boldsymbol{x}, \Pi; \theta)$ that will compute a score for each input/output pair, and which is linear in the parameter vector θ. Our prediction for query $\boldsymbol{x}$ will be the ranking $\hat{\Pi}$ which maximizes this score for a given input $\boldsymbol{x}$, *i.e.* $\hat{\Pi} = \arg\max_{\Pi} f(\boldsymbol{x}, \Pi; \theta)$. To learn the parameters θ we have to optimize the non-smooth loss function, therefore we define a convex upper bound on the loss given by

$$L(\boldsymbol{z}, \theta) = \max_{\Pi} \Big\{ \Delta(\Pi, \boldsymbol{z}) + f(\boldsymbol{x}, \Pi; \theta) \Big\} - f(\boldsymbol{x}, \tilde{\Pi}; \theta), \tag{1}$$

where $\tilde{\Pi}$ is any permutation that attains the minimum possible loss.[1] We sum the bound for M different training pairs $(\boldsymbol{x}_m, \boldsymbol{z}_m)$ to obtain a bound on the empirical estimate of the expected loss:

$$\frac{1}{M} \sum_{m=1}^{M} L(\boldsymbol{z}_m, \theta) = \sum_{m} \max_{\Pi} \Big\{ \Delta(\Pi, \boldsymbol{z}_m) + f(\boldsymbol{x}_m, \Pi; \theta) \Big\} - f(\boldsymbol{x}_m, \tilde{\Pi}_m; \theta). \tag{2}$$

To minimize this objective function, a variety of optimization methods can be used, including subgradient methods, and cutting plane methods to solve a constraint optimization formulation of the problem by introducing slack variables which introduce a linear constraint for each possible output. All these methods involve solving for a given θ the loss-augmented prediction problem, *i.e.* finding

$$\Pi^* = \arg\max_{\Pi} \Big\{ \Delta(\Pi, \boldsymbol{z}) + f(\boldsymbol{x}, \Pi; \theta) \Big\}. \tag{3}$$

It is thus critical that this problem can be solved efficiently for the choice of loss Δ and score f.

Quadratic score functions for ranking problems We now define quadratic score functions with terms that depend on the positions of two items in the ranking, and that can be written as the sum of a quadratic and a linear form of the permutation matrix:

$$f(\boldsymbol{x}, \Pi) = f_l(\boldsymbol{x}, \Pi) + f_q(\boldsymbol{x}, \Pi), \tag{4}$$

$$f_l(\boldsymbol{x}, \Pi) = \textstyle\sum_i s_i c_{\pi_i} = \boldsymbol{s}^\top \Pi \boldsymbol{c}, \qquad f_q(\boldsymbol{x}, \Pi) = \textstyle\sum_{ij} f_{ij} d_{\pi_i, \pi_j} = \mathrm{Tr}\{F \Pi D \Pi^\top\}, \tag{5}$$

where we used $\pi_i = j$ to denote that item i is ranked at position j, *i.e.* $\Pi_{ij} = 1$, we have dropped θ for clarity. In the linear term, the s_i denote the per item scores, which are linear in the parameter vector θ, and $\boldsymbol{c}$ is a design vector typically with non-increasing elements that encourages items with a high score to be ranked high. In the quadratic term, where the f_{ij} are parameters of the model, and D is a design matrix. For example, we can define $d_{kl} = |k - l|$ as the distance between two positions in the ranking. The quadratic term will then encode an incentive to put elements i and j close to each other in the ranking if $f_{ij} < 0$, and to put them far away in the ranking if $f_{ij} > 0$.

Finding the maximizer of such a score function, takes the form of a quadratic assignment problem (QAP). Since in general QAPs are NP-hard, the above formulation is of limited practical importance, unless approximate solutions are accepted, or additional structure is imposed on the quadratic terms.

Polynomial-time inference in c-star models Specific classes of QAPs are known to be solvable in polynomial time. One such class, identified in [11], is the class of problems where the interactions are restricted to form a c-star graph. That is, if there is a permutation matrix Π such that in $\Pi F \Pi^\top$ all non-zero elements appear in the first C rows and columns. In this case, we can solve the QAP as follows. For given positions in the ranking of the C elements in the core of the star, the remaining items have no interactions among each other. The remaining optimization problem is a linear assignment problem (LAP), which can be solved in $O(N^3)$, for N elements to rank. As there are $\binom{N}{C} C!$ placements of the C core elements, the original QAP can be solved in $O(N^{C+3})$.

If we use a linear loss function, then in the loss-augmented inference problem only a term linear in Π is added, which thus leaves the structure in the quadratic terms intact. Therefore, for linear loss functions, both loss-augmented inference and prediction can be done in $O(N^{C+3})$ time. This is much better than the general NP-hard case, but still very challenging in practice for large N. In the next section we show that for the specific case of p@k loss, and an appropriate definition of the pairwise terms, we can obtain a much more efficient approach that scales linearly in N.

[1] Throughout we use $\tilde{\Pi}$ and $\tilde{\Pi}^*$ to refer to permutations attaining the minimum loss for a given ground-truth.

4 Learning quadratic score functions for p@k

Before defining our model, we first observe that the precision-at-k loss is only sensitive to which elements are in the top k, not to how these are ordered, nor to how elements outside the top k are ordered. Therefore, as far as the loss is concerned, instead of considering the set of all permutations as our output space, we restrict ourselves to the set of all subsets of size k as our output space.

Score function for precision at k We will now design a score function that exhibits the same invariances as the loss, since there is little interest in differentiating among outputs that are equivalent w.r.t. the loss function. The linear term $f_l(\boldsymbol{x}, \Pi) = \boldsymbol{s}^\top \Pi \boldsymbol{c}$ is designed to score highest for selecting k elements with highest score, and to be invariant for permutations among each other. This can be achieved by defining $\boldsymbol{c}$ so that $c_i = 1$ for $1 \leq i \leq k$, and $c_i = 0$ otherwise. Since $\boldsymbol{c}$ is constant in the first k elements, it is invariant to permutations among these, and similar for the last $N - k$ elements.

For the quadratic term to be invariant for permutations among the top k elements, and among elements not in the top k, the matrix D must have a corresponding block structure where all elements in the same block are equal: $\begin{pmatrix} D_1 & D_2 \\ D_3 & D_4 \end{pmatrix}$, where D_1 is a $k \times k$ matrix, and D_4 a $(n-k) \times (n-k)$ matrix. Subtracting a constant from all blocks, will not change the predictions, nor the bound on the loss function. Therefore, without loss of generality, we can set $D_4 = \mathbf{0}$. Similarly, without loss of generality, blocks D_2 and D_3 can be set to zero by adding constants to the score functions s_i, and we can set D_1 to contain only 1's. Using these definitions, and $\boldsymbol{t}$ to denote the indicator vector of which elements are in the top k, *i.e.* with $t_i = [\![\pi_i \leq k]\!]$, the score function is now simplified to:

$$f(\boldsymbol{x}, \Pi) = \sum_i t_i s_i + \sum_{i,j} t_i t_j f_{ij} = \boldsymbol{t}^\top \boldsymbol{s} + \boldsymbol{t}^\top F \boldsymbol{t}. \tag{6}$$

Efficient inference in c-star models We now invoke the observation from Section 3, that in c-star QAPs can be solved by solving a set of LAPs, one for each assignment of the core of the star. Using i to index over elements not in the core of the star, and c and c' to index over elements that are in the core, we can express the score function for a fixed assignment of the core as

$$f(\boldsymbol{x}, \Pi) = \sum_i t_i \Big(s_i + \sum_c t_c f_{ic} \Big) + \sum_c t_c s_c + \sum_{c,c'} t_c t_{c'} f_{cc'}. \tag{7}$$

For fixed assignments of the core, the last two terms are constant. Therefore, the score is maximized by selecting the elements i with the largest values of $s_i + \sum_c t_c f_{ic}$. The best overall subset of k items can be found by looping over the 2^C configurations of the core elements to be in the top k or not, and finding the best elements to add to the top k from the non-core elements. Selecting the k largest elements in a list of length N can be done in $O(N)$ [12], and sorting these in $O(k \log k)$. Therefore, the overall algorithm to find the best subset of k items has running time $O(2^C(N + k \log k))$. The same holds for loss augmented inference, since we can write $\Delta_{p@k}(\Pi, \boldsymbol{z}) = \langle \boldsymbol{t}, \mathbf{1} - \boldsymbol{z} \rangle / k$. By replacing the s_i with $s_i + (1 - z_i)/k$, the best subset is obtained in the same manner.

5 Application to image labeling

Dealing with ambiguous training data ground-truth In many supervised learning problems, there is a unique target prediction. When learning p@k models, however, this is not necessarily the case, for example for image labeling when an image does not have precisely k relevant labels. Therefore, in general there will be many sets of k labels that attain the minimum possible loss for a given image. With each of those sets we can construct an uppperbound on the loss using Eq. (1).

One way to address this is to realize that any output $\tilde{\Pi}$ that attains minimum loss (1) bounds the loss, the tightest bound for any θ is the minimum of these bounds, given by

$$L'(\boldsymbol{z}, \theta) = \max_{\Pi} \{ \Delta(\Pi, \boldsymbol{z}) + f(\boldsymbol{x}, \Pi) \} - f(\tilde{\Pi}^*, x), \quad \text{with } \tilde{\Pi}^* = \arg\max_{\tilde{\Pi}} f(\tilde{\Pi}, x). \tag{8}$$

Since this bound is no longer convex, local minima of this objective can be obtained using a variety of methods, such as sub-gradient descend, or CCCP.

Another way to bypass this issue would be to use a loss which has a unique minimal solution, and thus $\tilde{\Pi}^* = \tilde{\Pi}$. Such a loss is given by the precision at k, with k the number of relevant items *for this image*. This is also known as the "break-even precision" (BEP), note that the learning objective has changed as well. In our experiments we follow this approach and report results on BEP.

Gradient calculation and kernel representation In practice we use a sub-gradient descend method to learn the parameters, at each gradient evaluation we compute Π^*, and $\tilde{\Pi}^*$ to compute the gradient of L. Note that in this way we obtain the true gradient, except when either of the two maximizers is not unique, which is a set of measure zero in the parameter space.

Using $\boldsymbol{t}^*$ and $\tilde{\boldsymbol{t}}^*$ to denote the indicator vectors for items ranked in the top k according to Π^* and $\tilde{\Pi}^*$ respectively, the gradients of the loss-bound in Eq. (1) or Eq. (8) are easily found using the definition of the score function in Eq. (6):

$$\frac{1}{2}\frac{\partial L'}{\partial F} = \boldsymbol{t}^*\boldsymbol{t}^{*\top} - \tilde{\boldsymbol{t}}^*\tilde{\boldsymbol{t}}^{*\top}, \qquad \frac{\partial L'}{\partial s} = \boldsymbol{t}^* - \tilde{\boldsymbol{t}}^*. \tag{9}$$

Thus for an interaction parameter f_{ij} we obtain a gradient signal if i and j are both in the top k according to Π^* but not according to $\tilde{\Pi}^*$, or vice versa. Similarly, for the label scores s_i we obtain a gradient signal if Π^* and $\tilde{\Pi}^*$ do not agree on i being in the top k.

For linear score functions $s_i(\boldsymbol{x}) = v_i + \boldsymbol{w}_i^\top \boldsymbol{x}$ we see that the gradient w.r.t. the weight vectors $\boldsymbol{w}_i$ always lies in the span of the data vectors. Therefore, we can express the weight vector as a linear combination of the training data points: $\boldsymbol{w}_i = \sum_m \alpha_{im}\boldsymbol{x}_m$, and the score function can be expressed in the form of a kernel:

$$s_i(\boldsymbol{x}) = v_i + \boldsymbol{w}_i^\top \boldsymbol{x} = v_i + \sum_m \alpha_{im}\boldsymbol{x}_m^\top \boldsymbol{x} = v_i + \sum_m \alpha_{im} k(\boldsymbol{x}_m, \boldsymbol{x}). \tag{10}$$

The kernel can implement the dot-product in an arbitrary feature representation extracted from the image, *e.g.* based on popular bag-of-word image representations. Similarly, an ℓ_2 regularizer on the $\boldsymbol{w}_i$ vectors can now be expressed using the kernel expansion.

Experimental evaluation Next we present our experimental evaluation to learn the ranking of labels for an image. For this we use the training set of the data set used in the ImageCLEF Photo Annotation task in 2010 [13]. The images are labeled with 93 diverse concepts ranging from concepts about people (*e.g.* single person, male, baby), visual appearance (*e.g.* macro, motion blur), objects (*e.g.* car, airplane, fish) and scene descriptions (*e.g.* landscape, city life). The data set contains of 8000 images with their Flickr-tags, we split the data into five folds (±6400 train and ±1600 test images), and report results averaged over the folds. For the sake of clarity we omit standard deviations since they are small compared to the differences between prediction methods. On average there are about 12 labels per image, and 830 images per label. We use the same features as the system that won the ImageCLEF 2010 Photo Annotation task [14], and which was also been used in [9], a concatenation of the improved Fisher vector representation [15] computed over SIFT and color features, and a binary vector denoting the presence of the most common Flickr-tags. In the experiments we compare the results when training the model using a kernel based representation, and when using scores of 93 one vs. all trained SVM classifiers.

In the experiments we compare an independent model (so without label dependencies) to a model with a c-star of 5 nodes. The 5 nodes for the core where chosen to optimise for the joint mutual information, in a greedy fashion. In Table 1 we show the results of our models on *Break Even Point*-precision, the precision-at-k where k is the number of relevant labels for this specific image. We observe that using the kernel representation gives a small but marked improvement over using the pre-trained SVM scores, this might be because the SVMs were trained for accuracy and not for BEP. We also see that the c-star structured models improve BEP performance by about 1% to 1.5%.

To investigate the influence of the structured models more, we also evaluate in an interactive image labeling setting, as also explored in [9]. The systems selects interactively a label, it obtains the relevance of this label from a user (in this experiment the relevance of the label is obtained from the ground truth). To select the label to query to the user, we use an idea related to the entropy but adjusted for the max-margin framework. For each label we compute the maximum score that can be obtained when it is forced be inside or outside the top k. The difference between the two scores

Table 1: Break Even Point precision on *ImageCLEF'10* for the independent and structured models. The performance is measured without human labeling and after $Q = \{1, 5, 10\}$ questions.

	Independent				C-Star, C=5			
	Auto	Q1	Q5	Q10	Auto	Q1	Q5	Q10
Pre-Trained SVM scores	66.6	68.5	75.0	81.1	68.0	69.8	76.1	81.6
Kernel	67.8	69.7	76.1	82.1	68.6	70.6	77.2	83.3

relates to the confidence of the system that the label should be selected or not. Intuitively, if the two scores are far apart the model prefers a specific state of the label (low entropy), while if the scores are very close the model does not care (high entropy). For each image we select the label with the lowest score difference. The results of the interactive image labeling experiment are also shown in Table 1. It shows that the structured model consistently improves over the independent model.

6 Conclusion

In this paper we have considered the relation between structured ranking models and quadratic assignment problems. While quadratic assignment problems are in general NP hard, we have identified a subclass which is solvable in polynomial time. Using the c-star models, and a loss function related to the precision-at-k measure, the algorithm runs in $O(2^C(N + k \log k))$. We have experimentally shown the performance of our model on a diverse and challenging image labeling dataset, where we rank the labels of a vocabulary according to the relevance for a given image. The c-star models show a modest but consistent improvement over the model assuming independent labels.

References

[1] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS*, 2003.

[2] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large Margin Methods for Structured and Interdependent Output Variables. *JMLR*, 6:1453–1484, 2005.

[3] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A Support Vector Method for Optimizing Average Precision. In *SIGIR*, 2007.

[4] Q. Le, A. Smola, O. Chapelle, and C.-H. Teo. Optimization of ranking measures. *JMLR*, 1(2999):1–48, 2010.

[5] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. In *ECML*, 2010.

[6] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, UK, 2008.

[7] T. Joachims. A support vector method for multivariate performance measures. In *ICML*, 2005.

[8] M. Choi, J. Lim, A. Torralba, and A. Willsky. Exploiting hierarchical context on a large database of object categories. In *CVPR*, 2010.

[9] T. Mensink, J. Verbeek, and G. Csurka. Learning structured prediction models for interactive image labeling. In *CVPR*, 2011.

[10] J. Petterson and T. Caetano. Submodular multi-label learning. In *NIPS*, 2011.

[11] G. Erdogan. *Quadratic assignment problem: linearizations and polynomial time solvable cases.* PhD thesis, Bilkent University, 2006.

[12] M. Blum, R. Floyd, V. Pratt, R. Rivest, and R. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4), 1973.

[13] S. Nowak and M. Huiskes. New strategies for image annotation: Overview of the photo annotation task at ImageCLEF 2010. In *Working Notes of CLEF*, 2010.

[14] T. Mensink, G. Csurka, F. Perronnin, J. Sánchez, and J. Verbeek. LEAR and XRCE's participation to Visual Concept Detection Task - ImageCLEF. In *Workshop ImageCLEF*, 2010.

[15] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *ECCV*, 2010.