Online Learning for Parameter Selection in Large Scale Image Search

Mohamed Aly
Computational Vision Lab
Electrical Engineering, Caltech
Pasadena, CA 91125 USA *vision.caltech.edu*
malaa@caltech.edu

Abstract

We explore using online learning for selecting the best parameters of Bag of Words systems when searching large scale image collections. We study two algorithms for no regret online learning: Hedge algorithm that works in the full information setting, and Exp3 that works in the bandit setting. We use these algorithms for parameter selection in two scenarios: (a) using a training set to obtain weights for the different parameters, then either choosing the parameter setting with maximum weight or combining their results with weighted majority vote; (b) working fully online by selecting a parameter combination at every time step. We demonstrate the usefulness of online learning using experiments on four different real world datasets.

1. Introduction

Searching large scale collections of images has become an important application of machine vision. There are currently several smart phone applications that allow the user to take a photo and search a database of stored images e.g. Google Goggles[1], Snaptell[2], and Barnes and Noble[3] application. These image collections typically include images of book covers, CD/DVD covers, retail products, and buildings and landmark images. The ultimate goal is to identify the database image containing the object depicted in a probe image, e.g. an image of a book cover from a different view point and scale. The correct image can then be presented to the user, together with some revenue generating information, e.g. sponsor ads or referral links.

It has been shown that *Bag of Words* (BoW) approach provides acceptable performance with fast run time and low storage requirements [15, 16, 7, 12, 2, 5, 11]. They have been used in image retrieval settings [16, 12, 15], near duplicate detection [6, 8], and image clustering [2]. However, one hurdle developers face with this (and other) approaches is the abundance of different parameters that affect their performance. For BoW, these parameters include the dictionary size, dictionary type, histogram weighting, normalization, and distance function. The typical solution to this is to have a labeled dataset divided into training and test sets to tune the parameters and quantify their performance.

The motivation of this work is to do away with labeled datasets in the process of selecting the best parameters and just rely on the feedback of actual users of the system to tune these parameters and improve the performance gradually over time. We study how to use two well known algorithms for no-regret online learning, Hedge [9] and Exp3 [3], to tackle this problem. They both assume we are given

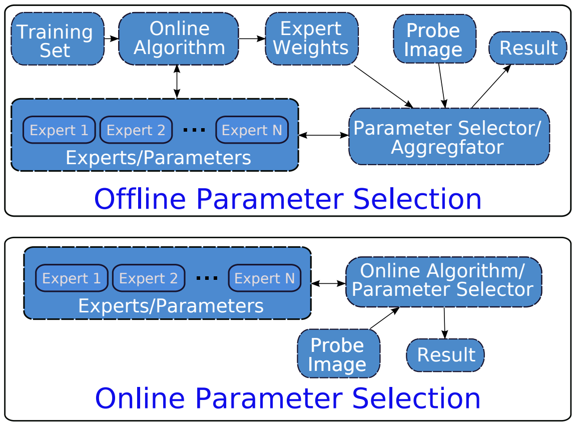


Figure 1: **Offline vs Online Parameter Selection.** In the offline case, the online algorithm is run with a training set (or user feedback in a training phase) to obtain fixed weights for the experts (parameters). Then, given a probe image, the parameter selector uses the outputs of the experts together with their weights to produce the final result. In the online case, the parameter selector works online by choosing the output of an expert given every input probe image. Experts are selected according to their past performance based on users feedback.

[1] tinyurl.com/yla655z
[2] tinyurl.com/582pq9
[3] tinyurl.com/mstn5b

a number of experts, from which we have to choose the output of one given every input pattern. The former works in the full information setting i.e. you not only know how well the chosen expert did, but you get to know how well all the other experts did. The latter works in the so-called bandit setting i.e. you only get to watch the result of the chosen expert. The idea of both is to have a weight assigned to every expert at any point, and choose one at random with probability proportional to that weight.

We apply these algorithms for parameter selection in two different scenarios:

1. We use the algorithms in an offline setting to obtain weights for different parameter settings of the BoW algorithm using the training set. Then we either choose the output of the maximally weighted expert or combine the experts' outputs with majority vote. The final output is evaluated on the test set.
2. We use the algorithms in an online setting, where given every input probe image, the algorithms have to decide which *expert* to choose and update their weights accordingly.

We make the following contributions:

1. We introduce no-regret online learning algorithms to the computer vision community and propose their use in the problem of large scale image search.
2. We show the usefulness of these algorithms for parameter selection in two settings: offline and online parameter selection. In particular, we show that with offline selection and weighted majority voting, we can outperform the single best expert. In addition, in the online scenario, we can achieve performance within 70-98% of the best expert.

2. No-regret Online Learning

We assume we have a collection of N experts $\{e_n\}_{n=1}^{N}$, each of which can predict the output of any given input. Input patterns are given sequentially, one at a time. Input p_t is given at time step t, and the algorithm chooses one of the N experts, $c_t = 1 \ldots N$, and output its prediction $e_{c_t}(p_t)$. Then the algorithm gets to watch the *loss* of this prediction $l_{c_t}^t$ and the predictions of all other experts l_n^t, $n = 1 \ldots N$, where $l_n^t \in [0,1]$. The goal of the algorithm is to minimize the loss of the chosen sequence of experts: $\sum_{t=1}^{T} l_{c_t}^t$. Define the *regret* of the sequence of choices made by the algorithm up to time T as the total loss of this sequence minus the loss of the single best expert (had we chosen it at every time step):

$$R_T = \sum_{t=1}^{T} l_{c_t}^t - \min_n \sum_{t=1}^{T} l_n^t$$

An algorithm has "no regret" if its regret is upper bounded by T i.e. $R_T = o(T)$. In that case, the average regret R_T/T

Algorithm 1 Hedge(ε)

```
Initialize $w_n^0 = 1$ for all $n = 1 \ldots N$
Loop $t = 1 \ldots T$
    1. Set $p_n^t = w_n^t / \sum_i w_i^t$
    2. Choose expert $c_t$ according to distribution $p^t$ and output its prediction $e_{c_t}(p_t)$
    3. Update weights $w_n^{t+1} = w_n^t (1-\varepsilon)^{l_{c_t}^t}$
```

Algorithm 2 Exp3(γ)

```
Initialize $w_n^0 = 1$ for all $n = 1 \ldots N$
Loop $t = 1 \ldots T$
    1. Set $p_n^t = (1-\gamma)\frac{w_n^t}{\sum_i w_i^t} + \frac{\gamma}{N}$
    2. Choose expert $c_t$ according to distribution $p^t$ and output its prediction $e_{c_t}(p_t)$
    3. Update weights $w_n^{t+1} = w_n^t \exp(\gamma r_n^t)$ where $r_n^t = \delta(c_t - n)\frac{\gamma}{N}\left(\frac{1-l_{c_t}}{p_n^t}\right)$ and $\delta(.)$ is the Kronecker delta
```

would decrease to zero as T approaches ∞. Intuitively this means that the algorithm has no regret if its incurred loss converges to the loss of the best possible expert available, so it does not regret making those choices.

We consider two well known no regret online algorithms:

1. **Hedge**: It works in the full information setting [9], i.e. we get to watch the loss of all experts at every iteration. It maintains a probability distribution over all the experts, from which it chooses one at random at every time t. The weights are then adjusted so that experts that suffered losses have their weights reduced, and those with no loss keep their weights unchanged, see alg. 1. The regret for Hedge is $R_T = O(\sqrt{T \ln N})$.
2. **Exp3**: It works in the bandit setting [3] i.e. we only get to watch the loss of the chosen expert $l_{c_t}^t$ at any time t. Like Hedge, it maintains a probability distribution over the experts, from which it chooses one at random at every iteration. Only the weight of the chosen expert is adjusted according to its loss, see alg. 2. The regret for Exp3 is $R_T = O\left(\sqrt{TN \ln N}\right)$.

3. Bag of Words (BoW)

BoW originated in text search applications [4] and has been successfully applied to various computer vision applications [15, 16, 7, 12, 2, 5, 11]. It is based on extracting local features from the images, e.g. SIFT [13], and then clustering them into *visual words*. Images are then represented as histogram counts of these visual words. An *inverted file* (IF) [18] is typically used for quickly searching through the stored image histograms. It is a data structure, in which for every visual word, a list of images that have this word is maintained, together with the count of that word in the images. Search is performed very quickly since only images that have overlapping visual words are considered [18].

BoW has a number of parameters that affect its recognition performance and run time:

Histogram Weighting:

1. none: use the raw histogram

2. binary: binarize the histogram i.e. just record whether the image has the visual word or not

3. tf-idf: weight the counts to decrease the influence of more common words and increase the influence of more distinctive words [18]

Histogram Normalization:

1. l_1: normalize so that they sum to one $\sum_i |h_i| = 1$

2. l_2: normalize so they have unit length $\sum_i h_i^2 = 1$

Distance Function:

1. l_1: use the sum of absolute differences i.e. $d_{l1}(h,g) = \sum_i |h_i - g_i|$

2. l_2: use the sum of squared differences i.e. $d_{l2}(h,g) = \sum_i (h_i - g_i)^2$

3. cos: use the dot product i.e. $d_{\cos}(h,g) = 2 - \sum_i h_i g_i$

Dictionary Type: The two leading methods to compute dictionaries:

1. Approximate K-Means (AKM): which approximates the nearest neighbor search within K-Means using a set of randomized Kd-trees [16].

2. Hierarchical K-Means (HKM): which builds a vocabulary tree by applying K-Means recursively [15] at each node in the tree.

Dictionary Size: It has been shown that having large dictionaries, on the order of tens of thousands of visual words, significantly increases the performance and search time in BoW with IF [16, 12].

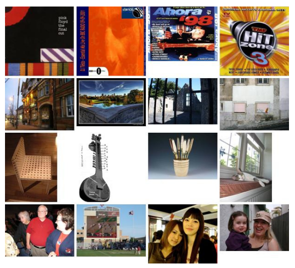
We consider five different promising combinations of histogram weighting/normalization/distance:

- *{tf-idf, l_2, cos}*: This is the standard way of computing nearest neighbors in IF [16, 17].
- *{bin, l_2, l_2}, {bin, l_1, l_1}*: The first was shown to work well with larger dictionaries in [11].
- *{none, l_1, l_1}, {none, l_2, l_2}*: The former is a novel combination using the l_1 distance, the later is equivalent to the standard one but with raw histograms.

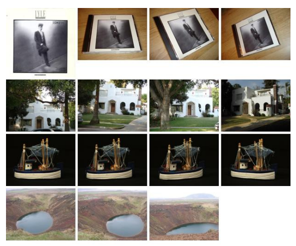
In addition to four combinations of dictionary type/size:

- *{AKM-10K, AKM-100K, AKM-1M}*: Approximate K-Means with 10K, 100K, and 1M visual words.
- *HKM-1M*: HKM with 1M visual words.

Therefore, we consider a total of 20 different parameter combination for BoW.



(a) **Example distractor images**. Each row depicts a different set: D1, D2, D3, and D4, respectively.



(b) **Example probe images**. Each row depicts a different set: P1, P2, P3, and P4, respectively. Each row shows a model image *(left)* with its probe images *(right)*

Figure 2: **Example Dataset Images**. See sec. 4.1.

Probe Sets			
	total	#model	#probe
P1	485	97	388
P2	750	125	525
P3	720	80	640
P4	957	233	724

Evaluation Sets		
Set	Distractor	Probe
1	D1	P1
2	D2	P2
3	D3	P3
4	D4	P4

Table 1: Probe Sets Properties and Evaluation Sets

4. Experimental Setup

4.1. Datasets

We have two kinds of datasets:

1. **Distractors**: images that constitute the bulk of the database to be searched.
2. **Probe**: labeled images, two types per object: **(a) Model Image**: the ground truth image to be retrieved for that object, **(b) Probe Images**: used for querying the database, representing the object in the model image from different view points, lighting conditions, scales, ... etc.

Distractor Datasets

- **D1: Caltech-Covers** A set of ~ 100K images of CD/DVD covers used in [1].
- **D2: Flickr-Buildings** A set of ~1M images of buildings collected from flickr.com
- **D3: Image-net** A set of ~400K images of “objects” collected from image-net.org, specifically images under synsets: instrument, furniture, and tools.
- **D4: Flickr-Geo** A set of ~1M geo-tagged images collected from flickr.com

Probe Sets

- **P1: CD Covers:** A set of 5×97=485 images of CD/DVD covers, used in [1]. The model images come from *freecovers.net* while the probe images come from the dataset used in [15].
- **P2: Pasadena Buildings** A set of 6×125=750 images of buildings around Pasadena, CA from [1]. The model image is image2 (frontal view in the afternoon), and the probe images are images taken at two different times of day from different viewpoints.
- **P3: ALOI** A set of 9×80=640 3D objects images from the ALOI collection [10] with different illuminations and view points. We use the first 80 objects, with the frontal view of each object as the model image, and four orientations and four illuminations as the probe images.
- **P4: INRIA Holidays** a set of 957 images, which forms a subset of images from [12], with groups of at least 3 images. There are 233 model images and 724 probe images. The first image in each group is the model image, and the rest are the probe images.

4.2. Setup

We used four different evaluation sets, where in each we use a specific distractor/probe set pair. Table 1 lists the evaluation sets used. Evaluation was done by choosing 100K images from the distractor set in addition to all the model images from the probe set. For every probe image, we get a ranked list of the images in the distractor + model sets, where highest ranked images are more likely to be the corresponding ground truth model image. Performance is measured as the percentage of probe images correctly matched to their ground truth model image i.e. whether the correct model image is the highest ranked image.

We want to emphasize the difference between the setup used here and the setup used in other "image retrieval"-like papers [12, 8, 16]. In our setup, we have only ONE correct ground truth image to be retrieved and several probe images, while in the other setting there are a number of images that are considered correct retrievals. Our setting is like the dual of image retrieval setting. In fact, in our probe sets, we invert the role of model and probe images e.g. P1 and P4 above.

We use SIFT [13] feature descriptors with hessian affine [14] feature detectors. We used the binary available from tinyurl.com/vgg123. Each evaluation set has its own sets of dictionaries, which are built using a random subset of 100k images of the corresponding distractor set. The probe sets were not included in the dictionary generation to avoid biasing the results. All experiments were performed on machines with Intel dual Quad-Core Xeon E5420 2.5GHz processor and 32GB of RAM. We implemented all the algorithms using Matlab and Mex/C++ scripts.

5. Experiments

Every one of our 20 parameter combinations represents a different expert, as defined in section 2. For each expert, we have the results for all probe images, see fig. 3. However, in the bandit setting we only use the output of the chosen expert. We use the two online algorithms in two different scenarios:

1. Offline Parameter Selection: where we divide the probe set randomly into training and test sets with 60% and 40% of the images respectively. We run the algorithm on the training set to obtain weights for the different experts. These weights are then used to combine the results of these experts using either the output of the highest weighted expert (see fig. 4) or weighted majority vote (see fig. 5-6) . The performance is then measured on the test set.

2. Online Parameter Selection: where we run the algorithms fully online on the whole probe set (see fig. 7-8). For Hedge, we use the information from all the experts, which in the actual setting would correspond to having the user rate the results of all the combinations. For Exp3, we use only the information of the chosen expert, whose results are presented to the user.

We measure the loss of the experts in two different ways:

1. Binary Loss: where we are only interested whether the expert returned the ground truth image as the top ranked result or not. We define $loss_{bin}(e,p) = 1 - \delta(g_e(p) - 1)$ where p is the probe image, e is the expert, and $g_e(p)$ is the rank of the ground truth image of probe p returned by expert e.

2. Rank Loss: where we use the rank information of the ground truth image in the experts results. We define $loss_{rank}(e,p) = \frac{g_e(p)-1}{M}$, where M is the maximum rank allowed ($M = 100$ in our experiments). If $g_e(p) = 1$, this corresponds to no loss.

6. Results and Conclusions

Fig. 3 shows the recognition performance of the 20 experts on the test sets for every evaluation set. We note that AKM-1M dictionary gives the best results, together with using binary histograms [11] or raw histograms and l_1 distance. We tried five different values {0.001, 0.01, 0.1, 0.25, 0.5} for ε and γ, the parameters of Hedge and Exp3 respectively.

6.1. Offline Parameter Selection

Figures 4-5 show the relative recognition with different settings for ε and γ. The performance value is divided by the performance of the single best expert. Fig. 4 shows results of choosing the expert with maximum weight. Fig. 5 shows results when using the weights to perform weighted majority vote for the outputs of the experts. We notice, in

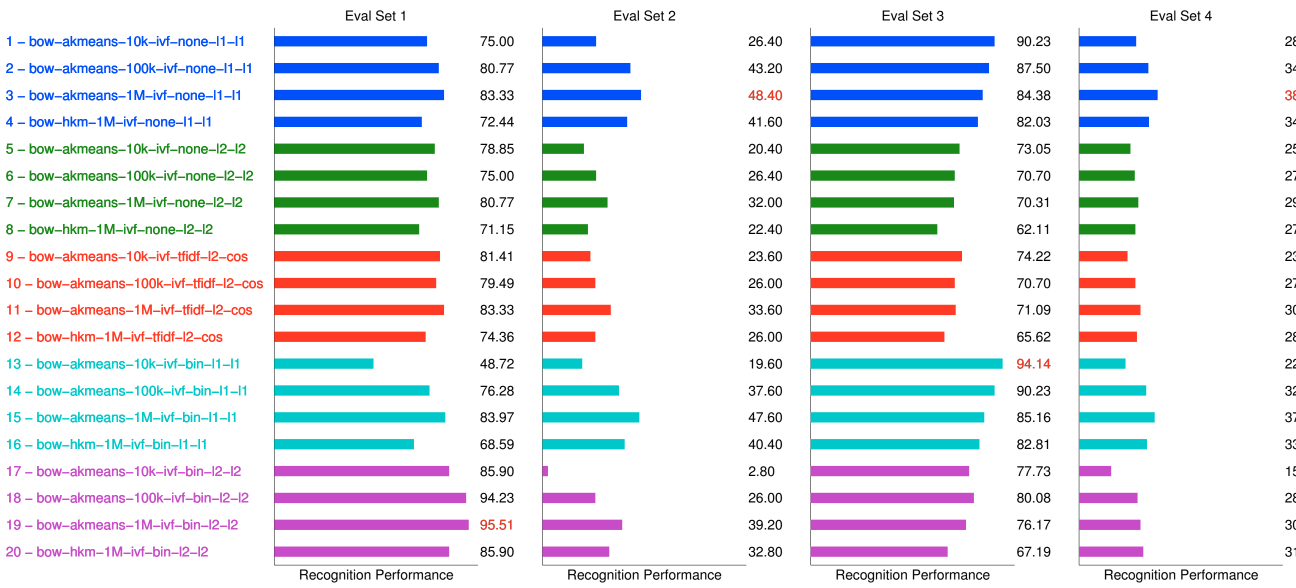


Figure 3: **Experts Test Recognition Performance.** Columns correspond to different evaluation sets (see table 1). The X-axis shows the recognition performance on the test set, while the Y-axis shows different experts corrsponding to different parameter combinations, see sec. 3. Each *{weight,normalization,distance}* combination has a different color. Text on the left shows the expert name and its number. Numbers in red are max performance.

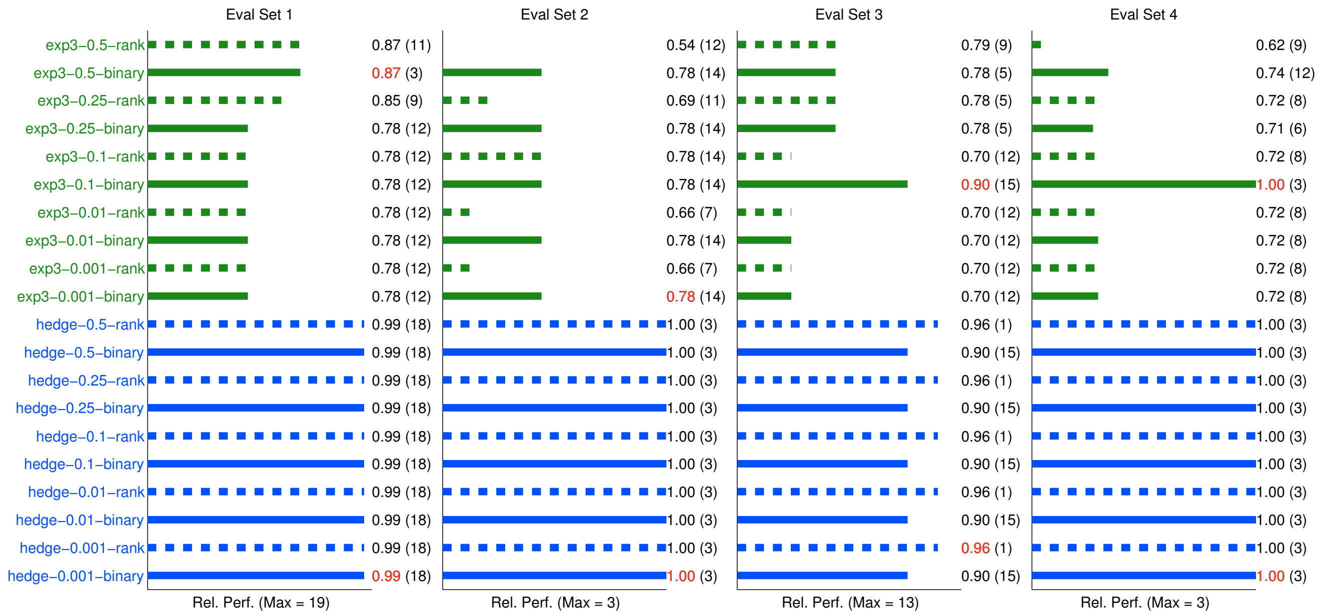


Figure 4: **Tuning Settings for Offline Selection with Single Expert.** Every column represents a different evaluation set. The X-axis represents the recognition performance on the test set divided by the max performance, while the Y-axis represents the different algorithms and loss functions, see sec. 5. Solid lines correspond to *binary loss*, while dashed lines correspond to *rank loss*. Blue corresponds to *Hedge*, and green represents *Exp3*. Numbers on the right are the relative performance, and those in red are the max for each algorithm. Number in parentheses represent the chosen expert with maximum weight, and the optimal expert is in the x-label *(bottom)*. Expert numbers are in fig. 3.

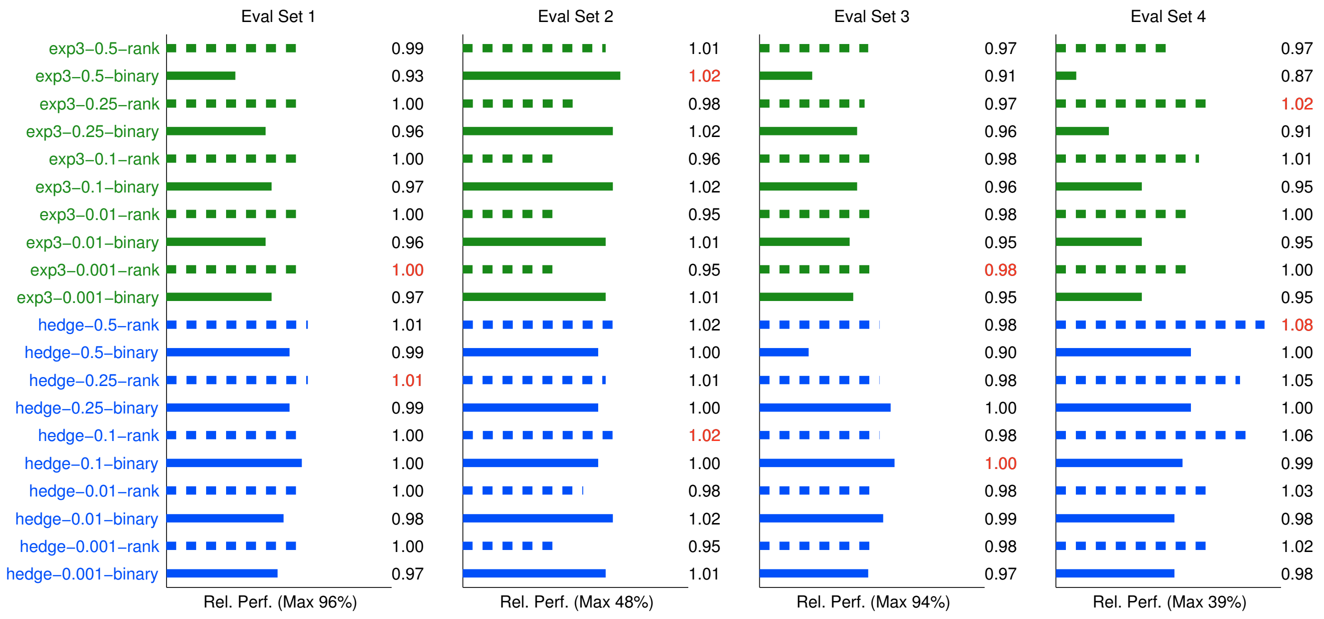
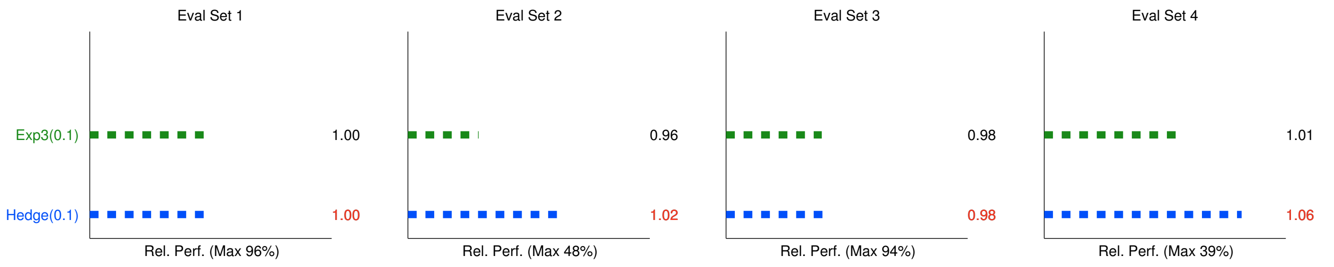
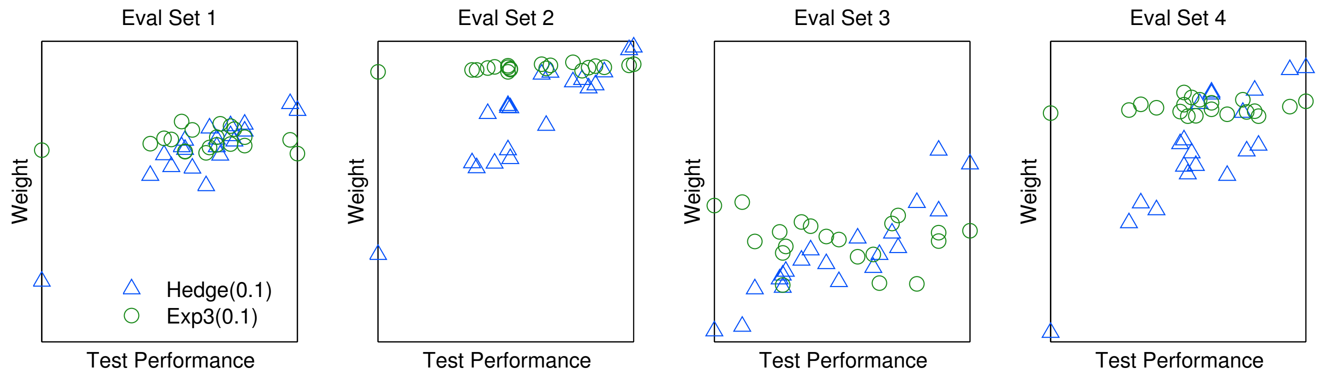


Figure 5: **Tuning Settings for Offline Selection with Weighted Majority.** Every column represents a different evaluation set. The X-axis represents the recognition performance on the test set divided by the max performance (in parentheses), while the Y-axis represents the different algorithms and loss functions, see sec. 5. Solid lines correspond to *binary loss*, while dashed lines correspond to *rank loss*. Blue corresponds to *Hedge*, and green represents *Exp3*. Numbers on the right are the relative performance, and those in red are the max for each algorithm. See section 6.1



(a) **Test set recognition performance with weighted majority vote.** Every column represents a different evaluation set. X-axis represents the recognition performance on the test set divided by the max performance (in parentheses), while Y-axis represents Hedge(0.1) & Exp(0.1) with rank loss function. Numbers in red are the max.



(b) **Correlation between expert weights and their performance.** Every column represents a different evaluation set. The X-axis shows the recognition performance on the test set, while the Y-axis shows the weight assigned from the training phase.

Figure 6: **Offline Parameter Selection Results**. See section 6.1.

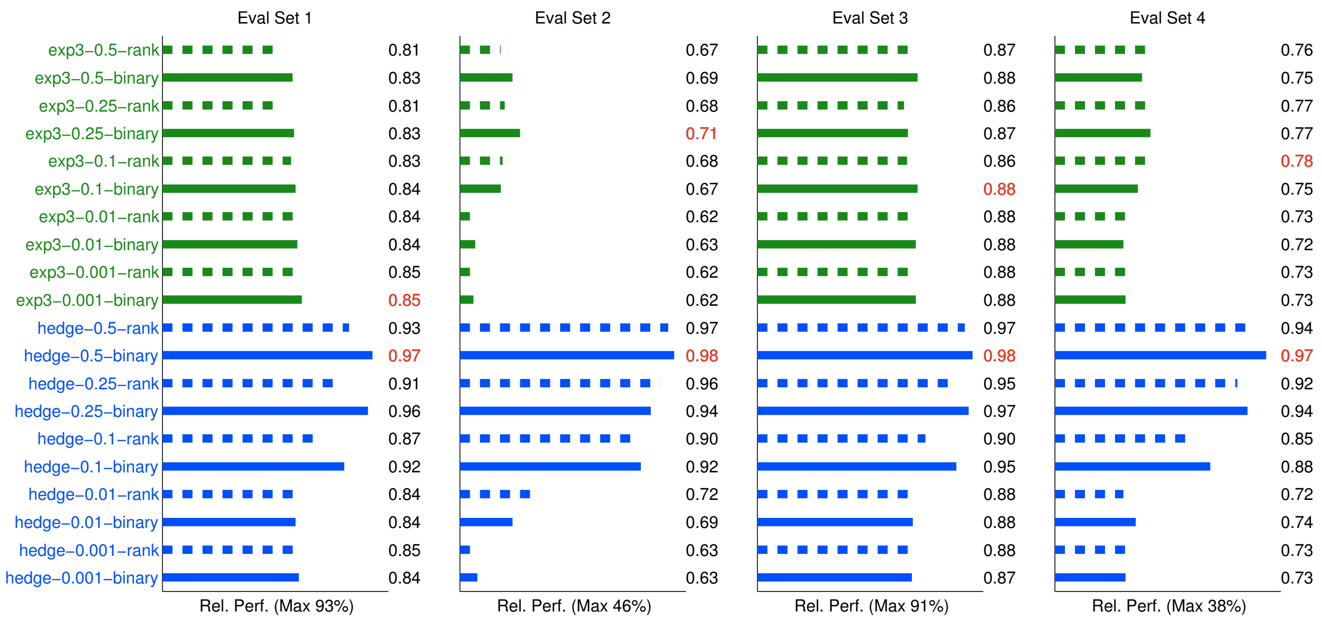
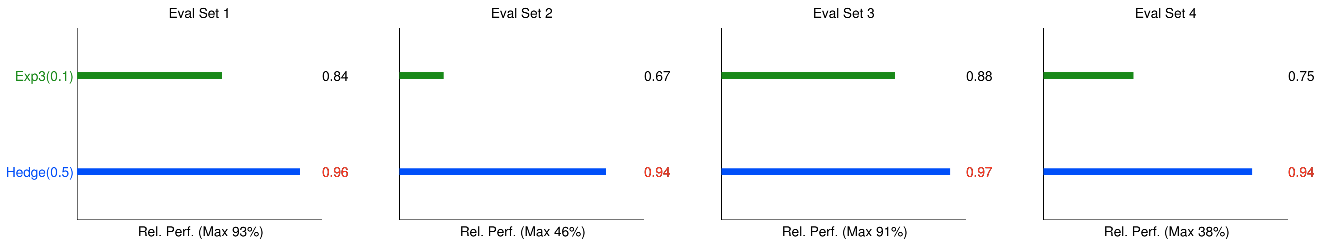
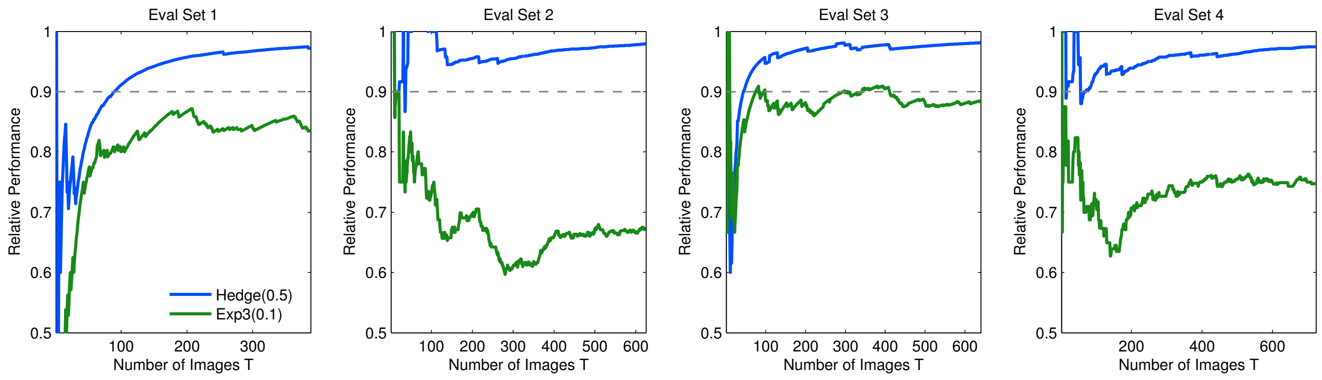


Figure 7: **Tuning Settings for Online Selection.** Every column represents a different evaluation set. The X-axis represents the online recognition performance divided by the max performance (in parentheses), while the Y-axis represents the different algorithms and loss functions , see sec. 5. Solid lines correspond to *binary loss*, while dashed lines correspond to *rank loss*. Blue corresponds to *Hedge*, and green represents *Exp3*. Numbers in red are the max for each algorithm. See section 6.2.



(a) **Online recognition performance on the full probe set.** Every column represents a different evaluation set. The X-axis represents the online recognition performance divided by the max performance (in parentheses), while the Y-axis represents Hedge(0.5) & Exp3(0.1) with binary loss function.



(b) **Online performance evolution over time.** Every column represents a different evaluation set. Y-axis shows the relative performance as a function of the number of probe images processed T on the X-axis for Hedge(0.5) & Exp3(0.1) with binary loss function.

Figure 8: **Online Parameter Selection Results**. See section 6.2.

this case, that most of them are comparable, and that using the rank loss function is generally better than using the binary loss function.

Figure 6 shows the experimental results for Hedge(0.1) and Exp3(0.1) (i.e. $\varepsilon = \gamma = 0.1$) with rank loss function. Fig. 6a shows recognition performance results on the test set with weighted majority vote. Fig. 6b plots the weights assigned to the experts versus their test performance. We notice the following:

- Using weighted majority vote outperforms choosing the expert with maximum weight. We usually get performance better than the best single expert, even in the bandit setting.
- With weighted majority vote, we get excellent performances, with the lowest performance at 96% of the best expert, and the highest performance exceeds the single best expert with 6%.
- Hedge is better than Exp3 with both weighted voting and when choosing maximum weighted expert, which is not surprising since Hedge uses all the information available.
- Hedge in general produces weights that are correlated with the test performance of the experts, unlike Exp3, which produces more evenly spread weights. This explains why Exp3 only selected the best expert in one evaluation set, while Hedge succeded in two. However, with weighted vote they are comparable.
- Using the rank loss function is generally better using the binary loss function.

6.2. Online Parameter Selection

Figure 7 shows the relative online recognition performance with different settings for ε and γ. The performance value is divided by the performance of the single best expert on the full probe set. We notice that Hedge(0.5) and Exp3(0.1) give the best performance, and that using the binary loss function is generally better than using the rank loss function.

Figure 8 shows the experimental results for Hedge(0.5) & Exp3(0.1) with binary loss function. Fig. 8a shows the online recognition performance on the full probe set (training + test sets). We plot the recognition performance divided by the performance of the single best combination. Fig. 8b shows how the relative online performance evolves over time by processing the probe images one by one. We note the following:

- Hedge is significantly better than Exp3. The performance of Hedge reaches within 94-97% of the single best performance, while Exp3 reaches 70-90% of that max. Again this is not surprising since Exp3 only gets partial information about the experts.
- Hedge(0.5) reaches within 90% of the best in only around 100 iterations.
- Using the binary loss function yields better results than using the rank loss function.

In summary, we have shown that no-regret online algorithms are quite useful for parameter selection in large scale image search. In the offline scenario, we can get performance better than the best expert, and in the online scenario the performance stays within 70-98% of the best.

Acknowledgements

This research was supported by ONR grant N00173-09-C-4005.

References

[1] Mohamed Aly, Peter Welinder, Mario Munich, and Pietro Perona. Scaling object recognition: Benchmark of current state of the art techniques. In *ICCV Workshop WS-LAVD*, 2009.

[2] Mohamed Aly, Peter Welinder, Mario Munich, and Pietro Perona. Towards Automated Large Scale Discovery of Image Families. In *CVPR Workshop on Internet Vision*, June 2009.

[3] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32:48–77, January 2003.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[5] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *CVPR*, 2009.

[6] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *CIVR*, pages 549–556, 2007.

[7] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*, 2007.

[8] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *British Machine Vision Conference*, 2008.

[9] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *journal of computer and system sciences*, 1997.

[10] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The amsterdam library of object images. *IJCV*, 61:103–112, 2005.

[11] H. Jégou, M. Douze, and C. Schmid. Packing bag-of-features. In *ICCV*, sep 2009.

[12] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

[13] David Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[14] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 2004.

[15] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. *CVPR*, 2006.

[16] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. *CVPR*, 2007.

[17] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[18] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, (2), 2006.