Universiteit Leiden

Computer Science

Learning Visual Concepts from Social Images

Zhenyang Li (S0937797)

Date: 01/08/2011

MASTER'S THESIS

Supervisor: Dr. Michael S. Lew

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Learning Visual Concepts from Social Images

Zhenyang Li

Abstract—Visual concept learning often requires a set of expert-labeled training images. In practice, nevertheless, acquiring a sufficient number of reliable annotations is laborious. A simple idea of learning a variety of semantic concepts from social tagged images thus appears as a nature way of replacing the expensive manual tagging. However, it is well known that user-contributed tags are ambiguous and subjective. This paper attempts to address the problem of learning visual concepts from weakly labeled social images. Intuitively, for a given concept, relevant images have to be emphasized more in the training process than irrelevant images. Starting from this, we first learn the visual relevance of social tags with respect to each training image by visual neighbor voting, and then incorporate the tag relevance into SVM and boosting classifiers in the form of importance weights. Our importance weighted classification is based on cost-sensitive learning, since high importance bears a high misclassification cost. Experimental results demonstrate the obvious improvements of the proposed methods, in particular for SVMs, in comparison to learning without using importance weights. Additionally, an empirical study on the impact of user tagging towards concept learning shows that, in general, better tagging accuracy leads to better performance, although some semantic concepts remain hard to learn in our experiments.

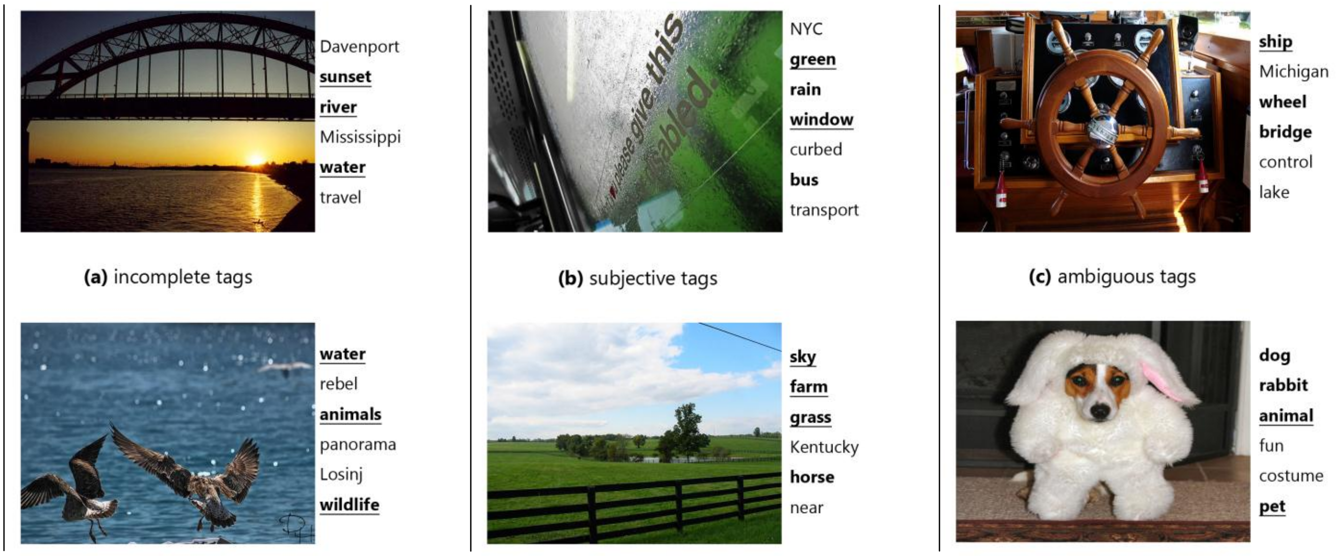
Index Terms—Visual concept learning, social images, social tagging, tag relevance learning, cost-sensitive learning, importance weighted classification.

1 Introduction

VISUAL concept learning is an important yet challenging problem in content-based multimedia information retrieval (CBMIR) areas [1]. It is fundamentally a classification task that determines whether an image or video shot is relevant to a given target concept. The semantic concepts can cover a wide range of topics such as those related to objects (e.g. car, lion), indoor and outdoor scenes (e.g. classroom, beach), events (e.g. parade, skiing), people etc. Automatically detecting these concepts helps in improving text-based (using only textual features) image or video retrieval, as well as complementing their manual annotations. However, how to effectively bridge the semantic gap between low-level visual features and high-level semantic concepts is still a key hindrance [2]. The performance of existing approaches can also be easily affected by the presence of intra-class variations, occlusion, background clutter, viewpoint and illumination changes in images and video clips [3]. In addition, another critical step along this task is the acquisition of sufficiently large amount of quality training data. It has been seen that large-scale data can directly benefit visual concept detection [7]. Rather than designing more intelligent classification algorithms and robust image features, we can simply use more data. The acquisition of reliable annotations, nevertheless, is a labor intensive process. For each concept to be learnt, training examples have to be annotated manually by expert annotators making these annotations expensive and limited. Labeling TRECVID 2010 dataset, for instance, requires collaborative annotation efforts from up to 47 research teams or organizations for 119,685 shots or keyframes with totally 130 concepts [5]. Such a tedious and costly manual labeling process will become extremely hard for the ultimate aim of annotating millions of images for thousands of visual concepts.

On the other hand, with the popularity of social media, there are increasingly large amounts of images and videos available on the web. For example, Flickr now hosts over 5 billion images with roughly 10 million new uploaded photos daily [4] and YouTube serves close to 3 billion video views per day with 48 hours of video uploaded every minute [6]. Apart from these rich multimedia databases, images and videos on the social networks are often accompanied by various forms of metadata like tags, ratings, comments, and EXIF information. These social context cues offer meaningful information about the content of multimedia and make it much easier to amass training data for visual concept learning. In particular, the user-contributed tags provide valuable source of descriptive information about the visual content of images and video shots. However, these social tags tend to be uncontrolled, ambiguous and overly personalized. For example, Fig. 1(a) includes two images in which the concept "bridge" and "bird" is obviously missing respectively. The photos in Fig. 1(b) are labeled with some subjective tags, such as "rain", "bus" or "horse". These concepts are not easy to get noticed in the photos. The concept "wheel" and "bridge", in the upper image of Fig. 1(c), are ambiguous, since "wheel" is commonly referred to as a circular object under a car or bus rather than the one used

- *Zhenyang Li is with Leiden Institute of Advanced Computer Science, Niels Bohrweg 1, 2333 CA, Leiden, The Netherlands.*
  Email: zhenyounglee@gmail.com
- *Michael S. Lew is with Leiden Institute of Advanced Computer Science, Niels Bohrweg 1, 2333 CA, Leiden, The Netherlands.*
- *Email: mlew@liacs.nl*



(a) incomplete tags

(b) subjective tags

(c) ambiguous tags

Fig. 1. Examples of social images with user-contributed tags. The tags in bold denote the ones we would consider their visual relevance with respect to the image content. In particular, the tags with underlines are thought of as truly relevant ones. It reveals three possible problems of social tagging: (a) incomplete tags, (b) subjective tags and (c) ambiguous tags.

to steer them, and the "bridge" here means the part of a ship where officers are controlling and steering the ship. The other photo in Fig. 1(c) shows a dog wearing a rabbit costume. It is somewhat confusing to annotate it with concept either "dog" or "rabbit". Automatically learning visual concepts from these weakly labeled web images thus appears as a nature way of replacing the expensive manual labeling. Some efforts on filtering or sampling the noisy tagged social images have been made in [8], [9]. But how such weakly labeled training examples affect visual concept learning in terms of user tagging accuracy, and compared with expert-labeled ones, is yet to be addressed.

In this paper, we empirically study on learning visual concepts from social images. First, we investigate two traditional algorithms, i.e. SVM and boosting, using multiple image features for visual concept learning. In particular, a common feature combination procedure is proposed to be integrated into different variants of the boosting algorithm. Second, in order to analyze social tagging, we present a visual neighbor voting model to learn the visual relevance of tags with respect to the image content. This model is inspired by recent successful tag relevance learning methods [10] [11] [12], that propagate the annotation tags of training images to a target image. We summarize their work by three weighting schemes, i.e. using uniform, distance-based and rank-based weights for each visually similar image, associated with a weighted nearest neighbor model. However, the main contribution of this paper is to introduce an importance weighted classification that incorporates the example-dependent importance weights into the learning frameworks of SVM and boosting classifiers. These importance weights are based on the tag relevance learned by visual neighbor voting, since more relevant example images have to be emphasized more in the training process for a given concept. Therefore, we aim to discriminate between different training examples by their importance weights in the classifier learning procedure using cost-sensitive learning techniques. Apart from this, all the proposed algorithms are evaluated by both the socially tagged and manually tagged images so as to explore the impact of the user-contributed tags, in terms of tagging accuracy, towards visual concept learning, and in comparison with manual annotations.

The remaining sections are organized as follows. Section 2 reviews some related works on visual concept learning and social image analysis. In section 3, we discuss the traditional SVM and boosting algorithms for visual concept learning using multiple image features. We then present a visual neighbor voting model to exploit the tag relevance of social images in Section 4. Section 5 describes our cost-sensitive learning problem and introduces the importance weighted extensions of SVM and boosting classifiers instead of directly learning visual concepts from weakly labeled social images. We setup experiments in Section 6 and the experimental results are presented in Section 7. Finally we conclude the paper in Section 8.

2 Related Work

2.1 Visual Concept Learning

The large-scale visual concept detection and annotation task (LS-VCDT) in ImageCLEF 2009 [13] used the MIR Flickr collection [14] as the benchmarking dataset. In total, 53 semantic concepts were evaluated and the team with the best results achieved an average AUC of 84% on their best run [15]. In their approach, they extract SIFT-like features encoded with "bag-of-words" model in different color spaces. Both salient point detector and dense grid are used for point sampling and in combination with spatial pyramid. The concept classifiers are trained using SVMs with χ^2 kernel.

Overall, the current state-of-the-art approaches in visual concept learning and annotation tasks are based on the "bag-of-words" model obtained by clustering of SIFT-like features. Within the "bag-of-words" representation, different point sampling strategies (e.g. keypoint detector or dense sampling), choices of descriptors (e.g. SIFT or SURF) and visual word assignment (e.g. hard or soft assignment) have also been studied. Specifically, salient point detectors, such as Laplace-of-Gaussian [16] and Harris-Laplace [17] based detectors, introduce robustness against viewpoint and illumination changes. Nowak et al. [18] showed that sampling on a regular dense grid in a uniform fashion consistently outperforms complex salient point methods in scene classification, since the more image patches are used the more of the appearance of an image can be captured. However salient points have the advantages of ignoring the homogenous areas in the image which is superior for object detection. SIFT [19] and SURF [20] are two commonly used local feature descriptors. Uijlings et al. [21] presented several improvements upon speeding up the calculation of densely sampled SIFT and SURF descriptors for real-time classification. Additionally, visual vocabularies can be created with k-means clustering or tree-based algorithms (e.g. Random Forests [22]). Each descriptor is typically assigned to a single predefined visual word. But it has been shown that assigning each descriptor to multiple visual words by using soft assignment is beneficial [23]. Beyond the "bag-of-words" model, Lazebnik et al. [24] proposed to use spatial pyramids of local features to encode a weak form of spatial information. It works by partitioning an image into increasingly fine sub-regions, and then the histograms of local features found inside each sub-region are computed and weighted according to their pyramid levels. Another idea is to construct a hierarchical organization of the visual vocabulary aiming to obtain more discriminative image representations. Spatial patterns of low-level visual words can be combined in to intermediate-level phrases or even sentences of visual words [25].

Support Vector Machine (SVM) classifier has been widely used for its outstanding performance and robustness against large feature vectors. The choice of kernel function is quite important to the classification performance. Zhang et al. [26] determined that in a "bag-of-words" approach to concept detection the earth movers distance and χ^2 kernel give the best accuracy and are to be preferred. Due to computational efficiency, Maji et al. [27] proposed an efficient classification method using SVMs with histogram intersection kernels. Boosting is another popular classification algorithm which has been successfully used for face recognition and object detection [28] [29]. This paper empirically compares the SVM and boosting classifiers in visual concept learning.

Moreover, Huiskes et al. [7] also pointed out that large-scale training data can directly benefit visual concept learning. Rather than designing more intelligent classification algorithms and robust image features, we can simply use more data. However, manually annotated image collections are usually size-limited due to the labor intensive process of manual labeling.

2.2 Social Tagging Analysis

The multimedia on the social networking websites (e.g. Flickr, YouTube and Facebook) are often companied by various forms of metadata, such as tags, ratings, comments, and EXIF information. While their strongly subjective nature, these rich social data make it much easier to amass training examples and have great potential to improve the performance of classic visual concept learning systems. As shown in [7] and [30], the social data, in particular, the user-contributed tags, can serve directly as image features for learning visual concepts. Particularly for the concepts that are difficult to learn with low-level visual features alone, the improvements are often considerable.

Despite the high popularity and advantages of social tagging, it is well known that tags provided by the grass-root Internet users are actually far from satisfactory as qualified descriptive indexing keywords for the visual content of the web images. The study in [31] revealed that user-provided tags are imprecise and only 50% of tags are related to the image content. Bischoff et al. [32] provided the tag distributions in three tagging environments. Their study indicated that there were a variety of user tagging motivations, such as opinion expression, self-presentation or attraction of attention. And only 45%-50% of tags can be used to enhance search experience. Aiming for improving tagging quality, an effort on tag refinement was made by Liu et al. [33]. They estimate the initial tag relevance scores based on probability density estimation and adopted random walk over a tag similarity graph to refine the relevance scores.

Another simple idea of learning visual relevance of the user-supplied tags with respect to the image content is based on the intuition that if users label visually similar images using the same tags, these tags are likely to reflect objective aspects of the visual content. Li et al. [11] and Verbeek et al. [30] proposed to propagate the annotation tags of training images to a target image by considering the presence of tags in its visual neighbors. Additionally, Ulges et al. [34] provided a probabilistic framework for detecting semantic concepts from weakly annotated training videos in the presence of irrelevant content. In their approach, the relevance of keyframes in the sequence is modeled as a latent random variable which is estimated during training. Therefore, we consider such exploitation of social tagging as a good starting point to aid visual concept learning.

3 Learning Visual Concepts

3.1 SVM

Support Vector Machine (SVM) algorithm constructs a hyperplane or set of hyperplanes in a high-dimensional space, which can be used for classification and regression analysis. It is a representation of the examples as points in space, mapped so that a good separation is achieved by the hyperplane that has the largest margin between the training data points of different classes, since in general the larger the margin the lower the generalization error of the classifier. Let $S = \{ (x_i, y_i) \mid i = 1, \cdots, N \} \subset \boldsymbol{R}^d \times \{-1, +1\}$ be the training samples, the two-class soft-margin SVM model which allows for misclassified examples works by solving the following optimization problem:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\left\|\mathbf{w}\right\|_{\mathcal{H}}^{2} + C\sum_{i=1}^{N}\xi_{i} \tag{1}$$

$$\text{s.t.} \quad y_{i}(\mathbf{w}\cdot\varphi(x_{i})+b) \geq 1-\xi_{i} \qquad \xi_{i} \geq 0 \tag{2}$$

The non-zero slack variable ξ_i expresses how much the example x_i fails to have the required margin, so it is introduced to measure the degree of misclassification in the optimization function (1). ξ_i takes a value greater than 1 if the corresponding training example lies to the wrong side of the decision boundary. Therefore, $\sum_i \xi_i$ indicates an upper bound on the total number of training errors. $C > 0$ is a regularization constant which determines the trade-off between the empirical risk and model complexity. By means of applying the kernel trick $\varphi : \boldsymbol{R}^d \to \mathcal{H}$, we can map the input data points into a higher-dimensional feature space $\mathcal{H}$ to create nonlinear classifiers. The classification decision function is defined as follows:

$$f(x) = \mathrm{sign}\left(\mathbf{w}\cdot\varphi(x)+b\right) \tag{3}$$

SVM is commonly regarded as a solid choice for classification. And many state-of-the-art visual concept detection systems achieved their best results by using SVM classifiers with χ^2 kernel [15] [26]. Recently, multiple kernel learning has been a topic of interest which associates image features with kernel functions and jointly learn the optimal combination of the kernels [35]. In this paper, we simply combine several kernels of multiple features into a single model by averaging their values. The RBF-based kernel function is used to measure the similarity between two images: $k(x_i, x_j) = \exp(-d(x_i, x_j)/\lambda)$, where $d(x_i, x_j)$ is the distance in a feature space between two images and λ is set as the average of all pair-wise distances among all the training images.

3.2 Boosting

Boosting is an ensemble learning framework to construct a strong classifier by combining a set of inaccurate classification rules (weak learners). We propose to use three variants of the boosting algorithm, including AdaBoost [36], RealBoost [37], and GentleBoost [37], for visual concept learning. Adaboost is the most commonly used version in which the weak learner directly outputs discrete class labels and the final classifier is defined to be a linear combination of the weak learners from each stage. While in RealBoost procedures, the weak learner produces a class probability estimate and its contribution to the final classifier is half the logit-transform of this probability estimate. Friedman et al. [37] also showed that boosting provides a generalized way to sequentially fit additive regression models of the form:

$$H(x) = \sum_{i=1}^{T} h_t(x) \tag{4}$$

where x is the input feature vector and T is the number of boosting rounds. $h_t(x)$ denotes a weak learner at each round t, and $H(x)$ is the final strong classifier learner. Thereby, they derive a "gentler" version called GentleBoost, which differs from RealBoost in that it takes adaptive Newton stepping rather than exact optimization at each stage and tends to put less weight on the outlier data points.

```
Input: Training set $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, where $y \in \{-1, +1\}$ is
the class label of example $x$, number of $M$ features, and number
of $T$ boosting rounds.
Initialization: Start with uniform weights $w_i = 1/N, i = 1, ..., N$.
for $t = \{1, \ldots, T\}$ do
  for $m = \{1, \ldots, M\}$ do
    (a) For feature $m$, fit the classifier $f_m(x) \in \{-1, +1\}$ using
    weights $w_i$ on the training examples.
    (b) Compute $\beta_m$ uniformly or according to (7) and (8).
  end for
  Get middle final weak classifier $h_t(x)$ according to (5) or (10).
  Compute $\epsilon_t = \mathrm{E}_w[I(y \neq h_t(x))]$, and $\alpha_t = \frac{1}{2}\log((1-\epsilon_t)/\epsilon_t)$.
  Update weights $w_i \leftarrow w_i e^{-\alpha_t y_i h_t(x_i)}$, $i = 1, \ldots, N$ and renormalize.
end for
Output: Final strong classifier: $H(x) = \mathrm{sgn}[\sum_{t=1}^{T} \alpha_t h_t(x)]$.
```

Fig. 2. AdaBoost Algorithm.

```
Input: Training set $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, where $y \in \{-1, +1\}$ is
the class label of example $x$, number of $M$ features, and number
of $T$ boosting rounds.
Initialization: Start with uniform weights $w_i = 1/N, i = 1, ..., N$.
for $t = \{1, \ldots, T\}$ do
  for $m = \{1, \ldots, M\}$ do
    (a) For feature $m$, fit the class probability estimate
    $p_m(x) = \hat{P}_w(y = +1 \mid x) \in [0,1]$ using weights $w_i$ on the train-
    ing examples.
    (b) Set $f_m(x) = \frac{1}{2}\log(p_m(x)/(1-p_m(x))) \in \boldsymbol{R}$.
    (c) Compute $\beta_m$ uniformly or according to (9).
  end for
  Get middle final weak classifier $h_t(x)$ according to (5) or (10).
  Update weights $w_i \leftarrow w_i e^{-y_i h_t(x_i)}$, $i = 1, \ldots, N$ and renormalize.
end for
Output: Final strong classifier: $H(x) = \mathrm{sgn}[\sum_{t=1}^{T} h_t(x)]$.
```

Fig. 3. RealBoost Algorithm.

In order to merge multiple visual features into our concept learning system, a feature combination procedure is introduced to be integrated at each round of these three boosting variants. The traditional boosting produces only one component weak classifier at each iteration. By contrast, at each round of our extension of the boosting procedures, several weak classifiers are trained on samples of each feature, and then combined into a single one (a middle final classifier) [38]:

$$h_t = \sum_{m=1}^{M} \beta_m f_m(x) \tag{5}$$

$$\text{s.t.} \quad \sum_{m=1}^{M} \beta_m = 1 \tag{6}$$

```
Input: Training set $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, where $y \in \{-1, +1\}$ is the class label of example $x$, number of $M$ features, and number of $T$ boosting rounds.
Initialization: Start with uniform weights $w_i = 1/N, i = 1, \ldots, N$.
for $t = \{1, \ldots, T\}$ do
    for $m = \{1, \ldots, M\}$ do
        (a) For feature $m$, fit the regression function $f_m(x)$ by weighted least-squares of $y_i$ to $x_i$ with weights $w_i$.
        (b) Compute $\beta_m$ uniformly or according to (9).
    end for
    Get middle final weak classifier $h_t(x)$ according to (5) or (10).
    Update class estimates $H(x_i) \leftarrow H(x_i) + h_t(x_i), i = 1, \ldots, N$
    Update weights $w_i \leftarrow w_i e^{-y_i h_t(x_i)}, i = 1, \ldots, N$ and renormalize.
end for
Output: Final strong classifier: $\mathrm{sgn}[H(x)] = \mathrm{sgn}[\sum_{t=1}^{T} h_t(x)]$.
```

Fig. 4. GentleBoost Algorithm.

where h_t denotes the final weak classifier at each round t, and f_m is the separately learned weak classifier using feature m. β_m indicates the linear combination weights, we can uniformly weight it by $\beta_m = \frac{1}{K}$. Yet another option for AdaBoost is to weight according to the same criteria of weighting the weak classifier at each round:

$$\epsilon_m = \mathrm{E}\left[w \cdot I(y \neq f_m(x)) \right] \tag{7}$$

$$\beta_m = \frac{1}{2} \log\left((1 - \epsilon_m) / \epsilon_m \right) \tag{8}$$

where $I(\cdot)$ denotes the indicator function which takes on the value 1 whenever the statement is true, and value 0 otherwise. w is the training weight for each example in boosting. Thus, these combination weights depend on the weighted training error rate of each weak classifier. Since RealBoost and GentleBoost use real-valued confidence-rated predictions rather than discrete positive or negative class labels $\{-1, +1\}$, a second weighting method of feature combination for them is defined based on generalization error (out-of-sample error rate):

$$\beta_m = \mathrm{E}\left[w \cdot \frac{1}{e^{-y f_m(x)}} \right] = \mathrm{E}\left[w \cdot e^{y f_m(x)} \right] \tag{9}$$

where the term $y f_m(x)$ indicates the margin, which is related to the generalization error. All the weights are normalized such that they sum up to 1, i.e., (6). In addition to linear combination, we also propose to select the best one, obtaining the largest combination weight, of all the weak classifiers trained on each feature as the final weak classifier at each round:

$$h_t = \arg\max_{f_{m \in M}} \beta_m \tag{10}$$

This way is much like a feature selection process. Our AdaBoost, RealBoost and GentleBoost algorithms are respectively described in Fig. 2, Fig. 3 and Fig. 4.

4 Exploiting Tag Relevance

4.1 Visual Neighbor Voting Model

A recent research topic on determining the visual relevance of the social tags has been studied in [11] [30] [39]. In general, the key idea is based on the nearest neighbor model that propagates the annotation tags of the visually most similar training images to a target image. Here, inspired by their work, we summarize it as a weighted nearest neighbor voting model: for each tag, a seed image will receive relevance votes from its visual neighbors which are labeled with this tag by users and the votes can be weighted according to their visual similarities. Fig. 5 illustrates an overview of this visual neighbor voting model without considering the contribution weight for each vote. Specifically, given an annotation concept w, its visual relevance r with respect to a seed image x_i is defined by taking a weighted sum of the votes from its K nearest neighboring images:

$$r(x_i, w) = \sum_{j=1}^{K} \pi_{ij} v(x_j, w) \tag{11}$$

$$v(x_j, w) = \begin{cases} 1 - \varepsilon, & \text{if } w \in x_j\text{'s tag list} \\ \varepsilon, & \text{otherwise} \end{cases} \tag{12}$$

where $v(x_j, w)$ indicates the vote from the neighbor image x_j, i.e., whether x_j is labeled with target concept w. And we use π_{ij} to denote the contribution weight when image x_j is voting on image x_i. The introduction of the non-negative constant ε (e.g. 10^{-5}) is a technicality to avoid zero prediction when none of the K nearest neighbor x_j is annotated with concept w. To ensure proper distribution and normalization so that $r \in (0, 1)$, we require that $\pi_{ij} > 0$ and $\sum_j \pi_{ij} = 1$.

The only parameter of this model is thereby π_{ij}, and we reasonably derive three weighting schemes for this weighted nearest neighbor model:

1. Uniform weighting: π_{ij} is equally weighted for all the visual neighbors.
2. Distance-based weighting: π_{ij} is weighted according to the measure of distance in the feature space between image x_i and neighboring image x_j.
3. Rank-based weighting: π_{ij} is weighted according to the ranking of image x_j among all the x_i's visual neighbors which are well ranked by their distance measure.

Based on these weighting approaches, below we present two effective tag relevance learning models driven by diverse features in an unsupervised or supervised manner.

4.2 Unsupervised Tag Relevance Learning

In order to seek a generic and unsupervised tag relevance learning model using the weighted neighbor voting strategy, we employ the uniform weighting scheme for all the visual neighbors, as well as the multiple feature learners [11] [39]. Specifically, we first perform tag relevance learning by searching for the nearest neighbors using each feature measure. Then, several base learners trained under different feature measures are combined in an uniform manner, since we have no prior knowledge of which base learner is most appropriate for a given target tag. Assume we just consider the K nearest neighbors of the seed image x_i. Since each visual neighbor will be

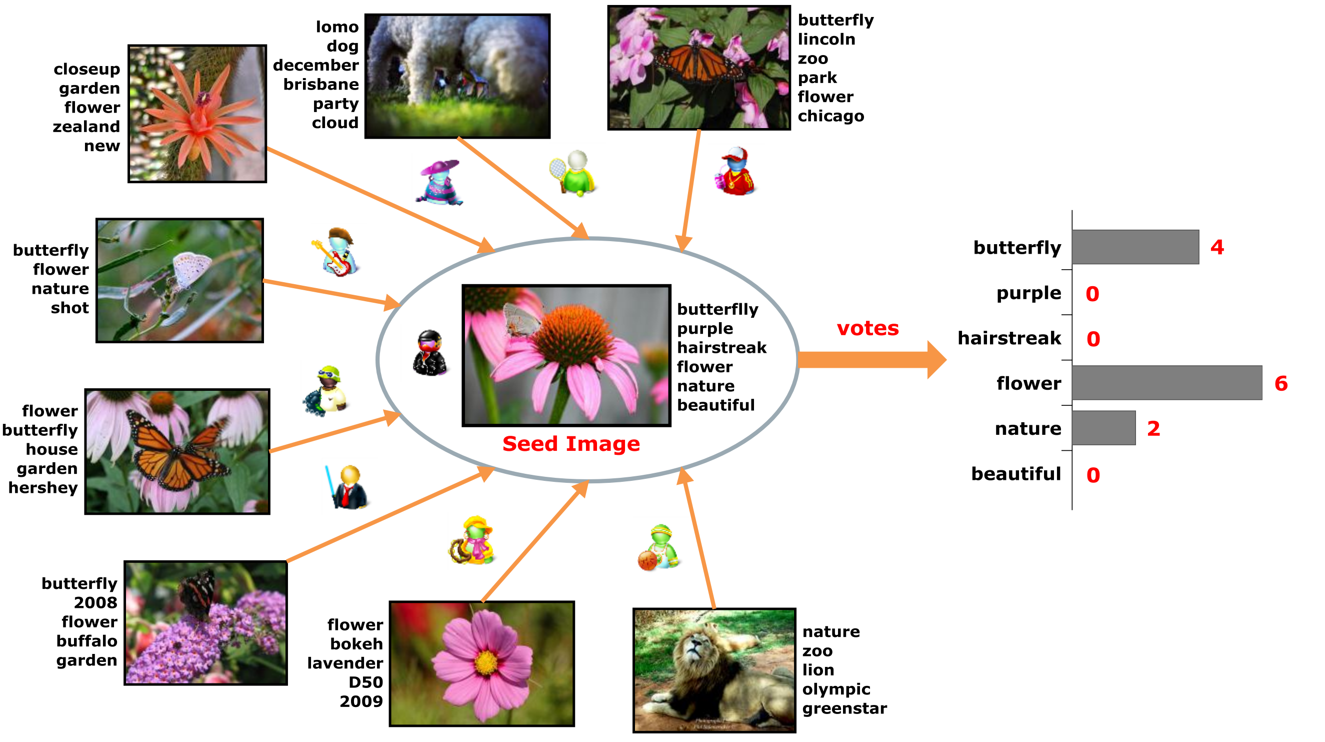


Fig. 5. Visual neighbor voting model. The tag relevance with respect to the visual content of an image is modeled by accumulating the neighbor votes received from visually most similar images of the seed image. For example, since 4 neighboring images are annotated with concept "butterfly", the seed image will obtain 4 votes for its tag relevance estimation. Moreover, if we consider to recommend new tags for the seed image, the concept "garden" would be preferred, because the accumulated neighbor votes for it is 3.

weighted by $\pi_{ij} = \frac{1}{K}$, the model (11) can be inferred as:

$$r(x_i, w) = \sum_{j=1}^{K} \frac{1}{K} v(x_j, w) \tag{13}$$

However, tags occurring frequently in the training image collection may dominate the results. To restrain such effects, we take into account the tag's prior frequency to estimate its prior probability [11]. Concretely, the prior probability for a given concept w is approximated as:

$$p_{\text{prior}}(w) = \frac{N_w}{N} \tag{14}$$

where N_w is the number of training images tagged with concept w, and N denotes the size of the entire training set. In general, the more neighboring images annotated with the target concept, the larger the tag relevance value would be. In the meanwhile, tags with high frequency are penalized for their high prior probabilities. As a result, we obtain the unsupervised tag relevance learning model using multiple features as follows [39]:

$$r_m(x_i, w) = \sum_{j=1}^{K} \frac{1}{K} v_m(x_j, w) - \frac{N_w}{N} \tag{15}$$

$$r(x_i, w) = \frac{1}{M} \sum_{m=1}^{M} r_m(x_i, w) \tag{16}$$

where r_m is the tag relevance learner trained using feature m. Note that function (15) does not necessarily obtain positive results, so in practice we set the minimum value ϕ, a very small constant (e.g. 10^{-5}), to avoid negative results in our experiments.

4.3 Supervised Tag Relevance Learning

When manually-labeled training images of given tags are available, the weighting parameter π_{ij} can be optimized to fit the tag relevance function. To this end, we employ two supervised tag relevance learning methods by performing distance-based and rank-based weighting. We follow the method proposed in [30], maximizing the log-likelihood of the tag relevance predictions for training images. The objective function is defined as follows:

$$\mathcal{L} = \sum_{i,w} \mu_{iw} \log\left(r'(x_i, w)\right) \tag{17}$$

Take care that if the annotation concept w is visually relevant to an image x_i, we aim to maximize its tag relevance $r' = r$, however, $r' = 1 - r$ should be maximized if concept w is irrelevant to image x_i. And μ_{iw} is the bias cost that takes into account the imbalance between concept presence and absence. Indeed, in practice, there are much more tag absences than presences, and absences are often much noisier than presences. This is because even if most concepts in annotations are relevant, the annotation often does not include all relevant concepts. We set $\mu_{iw} = 1/N^{+}$ if concept w is relevant, where N^{+} is the total number of positive training examples, and likewise $\mu_{iw} = 1/N^{-}$ when irrelevant, where N^{-} is the number of negative examples.

To define the weights directly as a function of the distance or rank metric, we use the weighting function introduced in [30] which was defined for distance-based weights, and here we also apply it to getting rank-based weights:

$$\pi_{ij} = \frac{\exp\left(-d_{\boldsymbol{\theta}}(x_i, x_j)\right)}{\sum_{j'} \exp\left(-d_{\boldsymbol{\theta}}(x_i, x_{j'})\right)} \tag{18}$$

where $d_{\boldsymbol{\theta}}$ is a distance or rank metric with parameter $\boldsymbol{\theta}$ that we want to optimize. Therefore, the weights π_{ij} decay exponentially with the distance or rank metric. Here we use linear combination for $d_{\boldsymbol{\theta}}(x_i, x_j) = \boldsymbol{\theta}^T \boldsymbol{d}_{ij}$, where $\boldsymbol{d}_{ij}$ is a vector of all base distances between image x_i and image x_j, or a vector of ranks for image x_j among the K nearest neighbors of image x_i under each distance measure, and the parameter $\boldsymbol{\theta} = (\theta_1, \cdots \theta_M)$ contains the positive coefficients of the linear distance or rank combination.

As we mentioned above, the weighted nearest neighbor voting model (11) tends to have relatively low recall scores for rare annotation keywords: in order to receive a high probability for the presence of a tag, it needs to be present among most visual neighbors with a significant weight. This, however, is unlikely to be the case for rare annotation terms. To overcome this problem, Verbeek et al. [30] introduced to perform concept-specific logistic transformation to boost the probability for rare concepts and decrease it for frequent ones. The logistic model uses the weighted neighbor voting predictions by defining:

$$r_{iw} = \sum_{j=1}^{K} \pi_{ij} v(x_j, w) \tag{19}$$

$$r(x_i, w) = \sigma(\alpha_w \cdot r_{iw} + \beta_w) \tag{20}$$

where $\sigma(z) = 1/(1+\exp(-z))$ is the sigmoid or logistic function and r_{iw} is the relevance estimation of concept w with respect to image x_i, which is learned by visual neighbor voting and using weighting function (18). This concept-specific model is equivalent to (11) up to an affine transformation. In practice, we estimate the parameters $\{\alpha_w, \beta_w\}$ and $\boldsymbol{\theta}$ in an alternating fashion.

5 Importance Weighted Concept Learning

Despite the high popularity and advantages of social tagging, it is well known that tags provided by the grassroot Internet users are actually far from satisfactory as qualified descriptive indexing keywords for the visual content of the web images. Therefore, in this section, two importance weighted concept learning algorithms are proposed to solve the problem of directly using noisy tags of social images for visual concept learning. Our approaches are inspired by current cost-sensitive learning techniques. First, we exploit the visual relevance of the tags that are present in the social images as shown in the previous section. Second, the tag relevance with respect to each training examples is integrated into the supervised learning process of SVM and boosting classifiers, in the form of importance weights.

5.1 Cost-sensitive Learning

The design of optimal classifiers with respect to losses that weight certain types of errors of training examples more heavily than others is denoted as cost-sensitive learning in machine learning and data mining communities. Classification problems such as fraud detection, medical diagnosis, or object detection in computer vision, are naturally cost sensitive. For example, in a face recognition based door locker system, the cost of mistakenly allowing an imposter to enter the house may be much higher than that of mistakenly rejecting a host, because the former kind of error would be a disaster and obviously much more serious than the latter.

Actually, the cost-sensitive learning process may involve many kinds of costs, such as test cost, teaching cost, intervention cost, etc., among which the most studied type is the misclassification cost [40]. Furthermore, the misclassification cost can also be categorized into two groups, i.e., problems with *class-dependent* cost [41] [42] [43] and *example-dependent* cost [44] [45]. In the former kind of problems, the cost is determined by error type, that is, misclassifying any example of a certain class into another class will always have the same cost, while misclassifying an example into different classes may result in different cost. In the latter kind of problems, the cost is determined by the example, while different examples may have different misclassification cost even when their error types are the same. Our work will focus on example-dependent cost-sensitive learning. Denote an example image x and its class label y. Given a set of examples $x \in X$ with class labels $y \in Y$ and $|Y| = L$, the traditional machine learning or classification methods try to generate a hypothesis $h: X \rightarrow \{1, \ldots, L\}$ minimizing the expected *misclassification error*:

$$\arg\min_h \mathrm{E}_{x,y}\left[I(h(x) \neq y)\right] \tag{21}$$

where we use $I(\cdot)$ to denote the indicator function which takes on the value 1 whenever the statement is true, and value 0 otherwise. Thus, these methods implicitly assume that the costs of all kinds of mistakes are the same. In our concern problem of example-dependent cost-sensitive learning, the general cost function $C_{h(x)} = C(x, y, h(x))$ specifies how much classification cost is incurred when an example x with correct label y is predicted to belong to class $h(x)$. Thereby it allows for cost dependence on each example x. We can also assume that the correct predictions are normalized so that $C_y = C(x, y, y) = 0$. Again, given a set of training examples $S = (x, \vec{C})^N$, where $\vec{C}$ is a vector of costs of misclassifying an example x as all possible labels, our goal is to find a classifier h which minimizes the expected *misclassification cost*:

$$\arg\min_h \mathrm{E}_{x,y,C}\left[C_{h(x)} \cdot I(h(x) \neq y)\right] \tag{22}$$

Our problem of learning visual concepts from weakly labeled social images can be viewed as a cost-sensitive learning problem, since for a given concept misclassifying a more relevant image should result in a higher cost than misclassifying an irrelevant image.

5.2 From Misclassification Cost to Importance Weight

Recently, a lot of work has attempted to convert machine learning algorithms and classification theory into cost-sensitive algorithms and theory. The research in this area falls mainly into three categories: 1) extending a particu-

TABLE 1
An Example of Cost Matrix for Binary Classification

	predict negative	predict positive
actual negative	c_{00}	c_{01}
actual positive	c_{10}	c_{11}

lar classifier learning algorithm so as to produce cost-sensitive generalizations; 2) using Bayes risk theory to assign each example to its lowest risk class; 3) making arbitrary classification algorithms into cost-sensitive ones. In particular, a general conversion proposed in [44] (and further study on multi-class case in [45]) is based on cost-proportionate weighting of the training examples, which can be realized either by feeding the weights to specific classification learners (e.g. boosting), or by carefully sub-sampling the training examples drawn from a weighted distribution. Rather than using "cost matrix" formulation which is more typical in cost-sensitive learning, they formulate example-dependent misclassification cost in the form of one importance weight per example and reduce this cost-sensitive learning problem into an importance weighted classification problem which can be solved very well by weighted rejection sampling techniques.

When the output space of the classification problem is binary, costs are associated with false negative and false positive, true negative and true positive predictions in the cost matrix formulation. Given an example and its cost matrix, only two entries, i.e. (false positive, true negative) or (true positive, false negative), are relevant for that example in the learning process, because it can only actually be either positive or negative example. Elkan et al. [42] and Zadrozny et al. [44] pointed out that these misclassification costs can be further reduced to one degree of freedom from a decision-making perspective: (false positive – true negative) or (false negative – true positive), which is the difference in cost between classifying an example incorrectly and correctly. For instance, consider the cost matrix in Table. 1, the cost difference we denote as example importance c here is defined as follows:

$$c = \begin{cases} c_{01} - c_{00}, & \text{if } y = -1 \\ c_{10} - c_{11}, & \text{if } y = +1 \end{cases} \tag{23}$$

This cost difference controls the importance of correct classification and just vary on an example-by-example basis. Then given a set of examples with the form (x, y, c), we aim to find a classifier h achieving the minimal importance weighted misclassification error:

$$\arg\min_{h} \mathrm{E}_{x,y,c}\left[c \cdot I(h(x) \neq y) \right] \tag{24}$$

An iterative weighting method was proposed for multi-class cost-sensitive learning problems in [45]. It also makes use of the importance weighted classification method, but critically differs from per-example formulation of the two-class cost-sensitive learning problem described above in that there is one classification cost associated with each possible prediction $h(x)$, whereas in the binary case there is a single importance weight associated with each example x. In order to take into account the different costs associated with multiple ways of misclassifying examples, they make a conversion by use of expanding data space. Specifically, given a set of examples consisting of $S = (x, \vec{C})$ of size N, where $\vec{C}$ is the cost vector specified above. The expanded data space S' of size NL, where $L = |Y|$ is the size of the class label set, is defined as follows:

$$S' = \left\{ (x, y, \max_{y'} C_{y'} - C_{y}) \mid \forall y \in Y \right\} \tag{25}$$

The importance weights given here, thereby, are more like benefits than costs, since larger costs will be mapped to smaller weights. However, as we adopt one-against-all strategy to solve multi-class classification problem using binary classifiers, in the following study we will focus on two-class cost-sensitive learning in which there is only one importance weight per example. How to further formulate this problem when the output space is out of binary is our future work and beyond the scope of this paper. Below, we will make an attempt to incorporate these importance weights into SVM and boosting classifier learning process, rather than employing resampling techniques, though it is more general and can be applied to arbitrary classifier learners.

5.3 Importance Weighted SVM

The problem of designing cost-sensitive extension to the SVM learning model has been studied in [46] [47] [48]. In addition to a general conversion by resampling, [46] proposed to shift the decision boundary by simply adjusting the threshold of the standard SVM classifier. This boundary movement method is obviously flawed when the data is non-separable, in which case cost-sensitivity requires a modification of both the separating hyperplane $\mathbf{w}$ and classifier threshold b. Another widely researched approach is to bias the penalties in the loss function [44] [47] [48]. It consists of introducing different penalty factors for different SVM slack variables of examples during training. Based on this idea, we modify the optimization formula (1) to incorporate the importance weights associated with each example:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\left\|\mathbf{w}\right\|_{\mathcal{H}}^{2} + C\sum_{i=1}^{N} c_i \cdot \xi_i \tag{26}$$

where c_i is the importance weight of example x_i and now regularization constant C controls model complexity versus importance weighted training errors. As shown in (26), the biased penalties method has direct effect on the support vectors of SVM classifier. However, it suffers from a flaw that it has limited ability to enforce cost-sensitivity when the training data points are separable, which is the opposite case of boundary movement method. Since, in practice, the training data is more likely to be non-separable, our implementation is based on the loss function (26) employing the biased penalties.

5.4 Importance Weighted Boosting

5.4.1 Importance Weighted AdaBoost

Various cost-sensitive extensions of AdaBoost algorithm are available in the literature, including AdaCost [49], CSB0, CSB1, CSB2 [50], and AdaC1, AdaC2, AdaC3 [51].

A simple idea to feed example importance weights to boosting procedures is to modify the initial boosting weights so as to break the importance symmetry. However, boosting re-updates all the weights at each iteration which may quickly destroy the initial asymmetry, and the predictor obtained after convergence usually makes little difference from that produced with symmetric initial conditions. Another natural heuristic is to modify the way of updating weights in the boosting procedures. Most of the previously proposed approaches [49] [50] [51] attempt to address this problem in AdaBoost, achieving cost-sensitivity by manipulation of its re-weighting mechanism and confidence parameters. AdaCost [49], for instance, introduces a cost adjustment function into weight updating rule of AdaBoost, aiming to increase the weight of a training example with higher importance "more" if it is misclassified, but decrease its weights "less" if otherwise. However the selection of the cost adjustment factor in AdaCost is ad-hoc and may easily induce poor performance [50]. Sun et al. [51] suggested a justified inference of weight updating parameter to maintain the boosting efficiency in reducing the weighted training error, while integrating the misclassification cost into the weight updating formula. Our importance weighted extensions of Adaboost are implemented using AdaC2 and AdaC3 algorithms [51], which respectively feed the importance weights to the weight updating rule of (27) and (28) at each round:

$$\alpha_t = \frac{1}{2}\log\frac{\sum_{i,\, y_i = h_t(x_i)} c_i w_i}{\sum_{i,\, y_i \neq h_t(x_i)} c_i w_i} \qquad (27)$$

$$w_i \leftarrow c_i w_i e^{-\alpha_t y_i h_t(x_i)},\ i = 1,\ldots,N$$

$$\alpha_t = \frac{1}{2}\log\frac{\sum_i c_i w_i + \sum_{i,\, y_i = h_t(x_i)} c_i^2 w_i - \sum_{i,\, y_i \neq h_t(x_i)} c_i^2 w_i}{\sum_i c_i w_i - \sum_{i,\, y_i = h_t(x_i)} c_i^2 w_i + \sum_{i,\, y_i \neq h_t(x_i)} c_i^2 w_i} \qquad (28)$$

$$w_i \leftarrow c_i w_i e^{-\alpha_t c_i y_i h_t(x_i)},\ i = 1,\ldots,N$$

where c_i denotes the importance weight for each example x_i. The weight updating function of AdaC2 or AdaC3, i.e. (27) or (28), will be equivalent to the weight updating function of original AdaBoost algorithm in Fig. 2, when the importance weight items are all set to 1.

5.4.2 Importance Weighted GentleBoost

As far as we know, there have been no cost-sensitive extension reported for GentleBoost in the literature. Furthermore, none of the weight manipulations in cost-sensitive AdaBoost can be easily applied to derive cost-sensitive extensions for other boosting variants, such as GentleBoost. Therefore, we next attempt to derive the importance weighted extensions for GentleBoost by following the formulation of the additive logistic regression mode [37].

Boosting provides a generalized way to sequentially fit an additive regression model (4) and it minimizes the following exponential cost function one term of the additive model at a time:

$$J(H) = \mathrm{E}\left[e^{-yH(x)}\right] \qquad (29)$$

where y denotes the class label $\{-1, +1\}$, and the term $yH(x)$ indicates the margin, which is related to the generalization error (out-of-sample error rate). This cost function can be thought of as a differentiable upper bound on the misclassification rate [52]. It also shows that $J(H)$ is minimized at:

$$H(x) = \frac{1}{2}\log\frac{P(y = +1 \mid x)}{P(y = -1 \mid x)} \qquad (30)$$

Hence we have $P(y = +1 \mid x) = \sigma(2H(x))$, where $\sigma(z) = 1 / (1 + \exp(-z))$ is the logistic or sigmoid function. This is equivalent to the usual logistic transform of $P(y = +1 \mid x)$ up to a factor 2. Boosting, consequently, can be viewed as step-wise estimation procedures for fitting an additive logistic regression model. In particular, GentleBoost optimizes $J(H)$ using adaptive Newton steps, which corresponds to minimizing a weighted squared error at each step. Specifically, at each round t, the function H is updated as $H(x) \leftarrow H(x) + h_t(x)$, where $h_t(x)$ take one Newton step to minimize a second order Taylor approximation of the cost function J:

$$\begin{aligned}\arg\min_{h_t} J(H + h_t) &= \arg\min_{h_t} \mathrm{E}\left[e^{-y(H(x)+h_t(x))}\right] \\ &\simeq \arg\min_{h_t} \mathrm{E}\left[e^{-yH(x)}(y - h_t(x))^2\right] \\ &= \arg\min_{h_t} \mathrm{E}\left[w \cdot (y - h_t(x))^2\right]\end{aligned} \qquad (31)$$

$$s.t. \qquad w = e^{-yH(x)} \qquad (32)$$

Replacing the expectation with empirical cost over training data, it reduces to minimizing the weighted squared error:

$$J_{wse} = \sum_{i=1}^{N} w_i (y_i - h_t(x_i))^2 \qquad (33)$$

where N is the number of training examples.

First, we propose to incorporate importance weight into the cost function formula as a linear factor:

$$J(H) = \mathrm{E}\left[c \cdot e^{-yH(x)}\right] \qquad (34)$$

where c denote the importance weight for each example x. Hence, we also choose to minimize the second order Taylor approximation of this new cost function:

$$\begin{aligned}\arg\min_{h_t} J(H + h_t) &= \arg\min_{h_t} \mathrm{E}\left[c \cdot e^{-y(H(x)+h_t(x))}\right] \\ &\simeq \arg\min_{h_t} \mathrm{E}\left[c \cdot e^{-yH(x)}(y - h_t(x))^2\right] \\ &= \arg\min_{h_t} \mathrm{E}\left[w \cdot (y - h_t(x))^2\right]\end{aligned} \qquad (35)$$

$$s.t. \qquad w = c \cdot e^{-yH(x)} \qquad (36)$$

Empirically, this also reduces to minimizing the weighted square error in (33), but with a new weight function (36). The weighs thus get updated by:

$$\begin{aligned} w^{(t+1)} &= ce^{-y(H(x)+h_{t+1}(x))} \\ &= ce^{-yH(x)} \cdot e^{-yh_{t+1}(x)} \\ &= w^{(t)} \cdot e^{-yh_{t+1}(x)} \end{aligned} \tag{37}$$

This is equivalent to initializing the boosting weights with importance weights, but updating them using the same rule in GentleBoost.

Compared with the exponential influence of the term $yH(x)$ which is associated with the generalization error, the importance weight c has much less effect on the cost function (34) as a linear factor. Therefore, a second heuristic idea is to formulate it inside the exponent of the cost function:

$$J(H) = \mathrm{E}\left[e^{-cyH(x)}\right] \tag{38}$$

Now the second order Taylor approximation we want to optimize is defined as follows:

$$\begin{aligned} \arg\min_{h_t} J(H + h_t) &= \arg\min_{h_t} \mathrm{E}\left[e^{-cy(H(x)+h_t(x))}\right] \\ &\simeq \arg\min_{h_t} \mathrm{E}\left[e^{-cyH(x)}(y - c \cdot h_t(x))^2\right] \end{aligned} \tag{39}$$

It then, empirically, reduces to minimizing the weighted squared error of the form:

$$J_{wse} = \sum_{i=1}^{N} w_i(y_i - c_i \cdot h_t(x_i))^2 \tag{40}$$

where $w_i = e^{-c_i y_i H(x_i)}$. However, in order to minimize the sum of squared residuals, the target value of $h_t(x_i)$ for each example x_i depends on its importance weight factor c_i. It makes no sense at all that the weak learner seeks different prediction ranges for different examples and we are not able to solve this problem by following the formulation of GentleBoost any more. Overall, our importance weighted Gentleboost is implemented according to the cost function (34), which only modifies the initialization procedure of the original GentleBoost in Fig .4.

5.5 Tag Relevance-based Importance Weighting

Since one-against-all strategy is performed to reduce our multi-class classification problem into multiple binary problems, two tag relevance-based importance weighting schemes are proposed, namely per-concept weighting and per-image weighting, concentrating on the binary distinction of positive vs. negative. In general, for a given concept, higher relevance value leads to a higher importance weight in the training process. And we assume that all tag relevance values are normalized into $(0, 1)$.

In per-concept weighting scheme, for each annotation concept, we first learn the visual relevance of this concept with respect to all the training images even if it is not present in the user-contributed tags of an image. Then, to solve the binary classification problem of a target concept, all the images labeled with this concept are trained as positive examples and take importance weights that equal to their tag relevance value, while images not labeled with this concept, as negative examples, take importance weights according to $(1 - \mathit{TagRelevance})$. On the other hand, for each training image, we only learn the relevance of all its user-provided tags in per-image weighting scheme. And then their tag relevance and importance weights are equivalently used regardless of an image is trained as positive or negative example in a binary classification problem of a given concept.

6 Experimental Setup

6.1 Dataset

Social20 [53] is a collection of 19,972 social tagged images with 20 diverse visual concepts randomly collected from Flickr. For each concept, it consists of 1000 example images labeled with that concept, as well as other annotation concepts, by user tagging. It has been known that social tags can be very subjective and overly personalized, as a result, often irrelevant to the visual contents of images. Therefore, these social images have also been manually relabeled in terms of their visual relevance: we consider a semantic concept and an image relevant if the concept is clearly visible in the image and we shall relate the concept to the visual content easily and consistently with common knowledge. Finally, only 5,241 images are preserved after the manual relabeling, because some of the images are visually irrelevant to all of our target 20 concepts. The dataset is evenly split into training data and testing data. In our experiments, we have used both social tags and manual tags to investigate our algorithms and the performance evaluation is always based on the manual annotations.

6.2 Evaluation Criteria

6.2.1 Image Ranking Evaluation

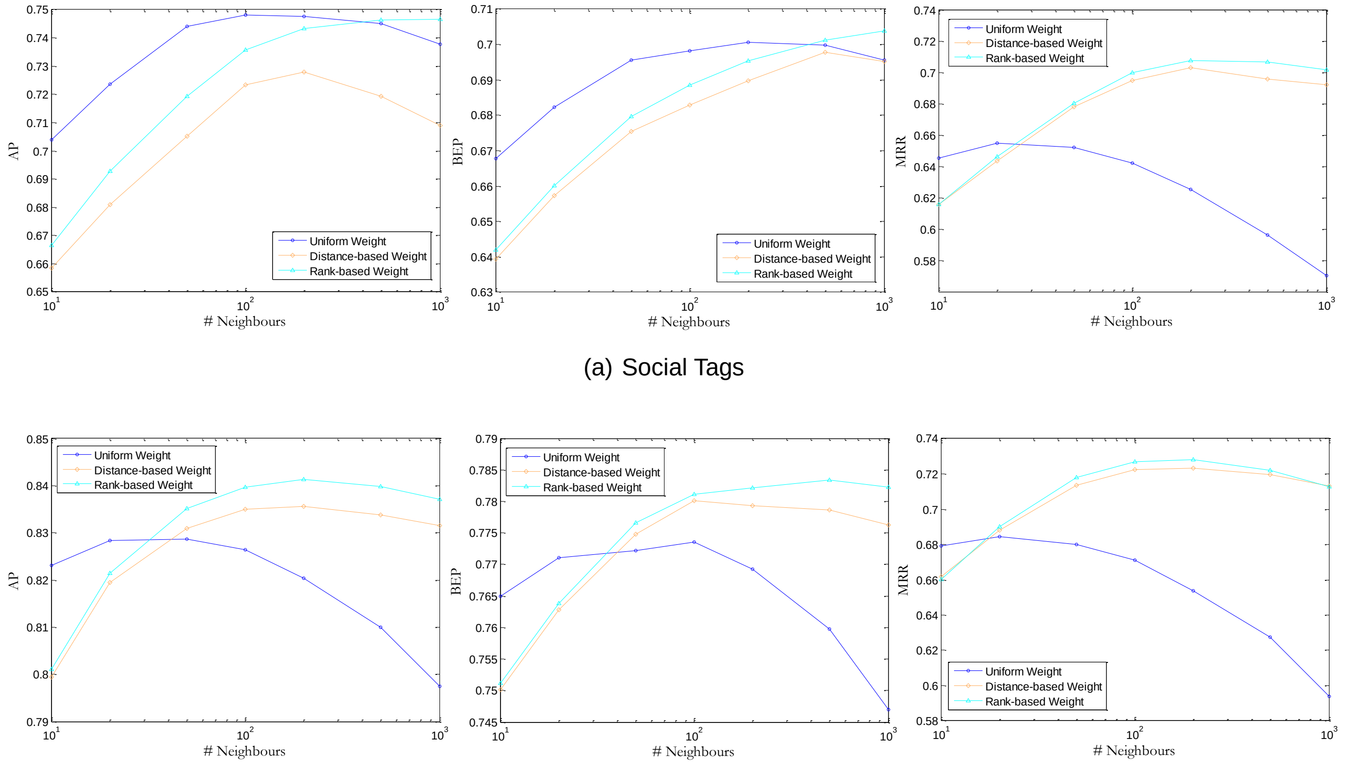
To measure image ranking performance we use average precision (AP) and break event point precision (BEP). For a given semantic concept, we rank all the images by their predicted probabilities and evaluate precisions at each position according to the manual annotations. AP averages the precision over all positions of relevant images, whereas BEP computes the precision just at one position, which is the number of relevant images that are manually labeled with that concept. Both measures are evaluated per concept, and finally averaged over all the concepts to obtain a single measure. These measures indicate how well we can retrieve relevant images from the database in response to the keyword-based user queries.

6.2.2 Concept Ranking Evaluation

In addition to image ranking measures, we also evaluate concept ranking performance by mean reciprocal rank (MRR). For each image, we rank all its possible concepts by their predictions, then compute mean of the reciprocal ranks of the manually annotated concepts for this image and finally average them over all the images. This measures how well we can automatically identify or recommend relevant annotation concepts for images.

6.3 Visual Feature Extraction

We extract global features and local features of images which are commonly used for image retrieval and categorization to enhance the performance of visual concept



(a) Social Tags

(b) Manual Tags

Fig. 6. Comparison in terms of AP, BEP and MRR performance of visual neighbor voting model using uniform weights, distance-based weights and rank-based weights. All the models are trained with different values for parameter K, as well as using (a) social tags or (b) manual tags. Note the log scale on the horizontal axis.

learning. There are two types of global visual features: Color and Gist. The Color features consist of the color correlogram [54], the texture moments [55] and the RGB color moments. The Gist is a popular global feature which represents the dominant spatial structure of a scene by a set of perceptual dimensions, such as naturalness, openness, roughness, expansion, ruggedness [56]. As for local features we use the SIFT descriptor [19], and both dense grid and Laplacian of Gaussian (LoG) keypoint detector are used for point sampling. Each local feature descriptor is quantized using k-means clustering (1000 cluster centers) on samples from the training set, and images are then represented as "bag-of-words" histograms. In order to encode the spatial layout of the image to some degree, we follow the approach of [24], and compute the histogram by two-level spatial pyramids over different image regions. The images are sampled over three horizontal sub-regions, i.e. 1x3, reflecting the typical top, middle and bottom layout of landscape photography. At last, two-level histograms are weighted and combined into a single histogram (4*1000-d). To compute distances from the feature descriptors in the visual neighbor voting model and SVMs kernel functions, we use Euclidean distance (L_2) for Color and Gist features, Chi-square distance (χ^2) for SIFT and Dense SIFT histograms.

7 Experimental Results

7.1 Experiment 1: Tag Relevance Learning

In our first set of experiments we use different variants of the visual neighbor voting model to predict the visual relevance of the target 20 annotated concepts. The tag relevance learning methods we evaluated include one unsupervised model using uniform weights and two supervised models using distance-based or rank-based weights. A common parameter to optimize for all these models is K, which is the number of visual neighbors used to vote a seed image. We test and choose K from the set $\{10, 20, 50, 100, 200, 500, 1000\}$. The supervised learning models, particularly, required to be trained on a set of manually labeled example images. This can be done either by using held-out data or in a leave-one-out manner. In our experiments we have used the same training set for neighbor voting and supervised learning in leave-one-out manner. Moreover, we investigate this visual neighbor voting model both by use of images with social tags and manual annotations. The social tagged dataset was filtered by relabeling and the resultant manually tagged dataset has a smaller size than the former. Because of the noise in the social tags, performance is always evaluated based on the manual annotations. In Fig. 6, we give an overview of performance of the three tag relevance learning methods in terms of AP, BEP and MRR, as a function of the number of visual neighbors K which is used in the nearest neighbor searching process.

TABLE 2
COMPARISON ON AP, BEP, AND MRR (%) FOR DIFFERENT BOOSTING VARIANTS

(a) Social Tags

	Ada(b)	Real(b)	Gentle(b)	Ada(u)	Real(u)	Gentle(u)	Ada(e)	Real(e)	Gentle(e)
AP	69.1	62.5	71.6	71.3	67.5	**71.9**	70.4	66.6	71.8
BEP	65.6	59.8	68.0	67.8	64.8	68.5	67.4	64.0	**68.6**
MRR	59.5	41.6	68.3	70.1	55.9	71.0	65.8	51.9	**71.3**

(b) Manual Tags

	Ada(b)	Real(b)	Gentle(b)	Ada(u)	Real(u)	Gentle(u)	Ada(e)	Real(e)	Gentle(e)
AP	81.8	76.3	83.4	84.6	81.3	**85.0**	83.1	80.2	84.7
BEP	75.7	71.3	77.3	78.6	75.8	**78.8**	77.3	74.9	78.6
MRR	53.2	47.6	69.8	61.9	64.5	72.6	53.7	62.1	**72.8**

(b), (u) and (e), respectively, denote selecting the best feature, uniform and error-based weighting scheme for feature combination at each round of boosting procedures. The better performance between boosting variants using each weighting scheme is underlined, while the best performance among all methods is ***bolded****.*

As shown in Fag. 6(a) when trained on social tagged images, all the variants of the weighted nearest neighbor voting model, i.e., using uniform, distance-based and rank-based weights, can make constant improvements in terms of AP, BEP and MRR performance with an increasing number of visual neighbors used for voting. Meanwhile, using much more neighbors has a slight negative effect on performance. This is easy to understand that it is more likely to include useful visual neighbors from more different neighborhoods, however, more neighbors will lead to more noise when most of the useful neighbors have been included. The optimal parameter setting for our three variant models is $K=100$, $K=200$, and $K=500$ respectively. In addition, we observe that the uniform and rank-based weighting models get very comparable results in terms of AP and BEP. But the MRR score evaluated using uniform weights drops significantly as a result of using more and more neighbors, and it is mostly much lower than that when using distance-based or rank-based weights. Therefore the supervised tag relevance learning model has much better discriminative capabilities between semantic concepts than the unsupervised learning model in this case. Using rank-based weights always yields higher values of AP, BEP and MRR than using distance-based weights.

The results of using manual annotations, in Fig. 6(b), illustrate a considerable performance improvement compared to using social tags. And the increase is more pronounced in terms of AP and BEP than in MRR. We can observe very similar impact of using an increasing number of visual neighbors on performance. However, the AP, BEP and MRR scores yielded by the uniform weighting model start to decrease quickly from the beginning with a relatively small value of parameter K. The optimal choice of K neighbors, in this case, is $K=50$, $K=200$ and $K=200$ respectively. The unsupervised tag relevance learning model now is largely outperformed by the supervised learning models in terms of all the evaluation criteria. Likewise, using rank-based weights achieves better performance than using distance-based weights.

For the following experiments, we also use these three tag relevance learning methods for comparisons with other visual concept learning algorithms, and the parameter of K neighbors is always set optimally.

7.2 Experiment 2: Visual Concept Learning

In this section we investigate SVMs, boosting variants, as well as their importance weighted extensions for visual concept learning by use of social tags or manual annotations. First, we evaluate different variants of the boosting algorithm, and feature combination approaches integrated at each round of boosting procedures. Second, an overall comparison of performance between SVMs, boosting variants and tag relevance learning methods is presented. At last, we analyse the results of our importance weighted SVMs and boosting algorithms when learning visual concepts from weakly labeled social images.

7.2.1 Evaluating Boosting Variants

We compare three boosting variants, including AdaBoost, RealBoost and GentleBoost. Three different feature combination approaches are also integrated into each boosting variant and evaluated. Moreover, we follow the AdaBoost.MH algorithm [37] to convert the multi-class problem using one-against-all strategy. However, rather than building one large tree using class label as an additional input feature, we implemented it using the more traditional direct approach of building separate trees to solve each binary problem. All the boosting variants used classification and regression tress (CART) as weak learners. The parameters of CART classifiers are optimally selected. Unless otherwise noted, we at most construct 100 trees for each feature, i.e., the maximal number of boosting rounds is 100.

From the results in Table 2 we can make several observations. For both choices of using social tags and manual tags, GentleBoost achieves the best performance among all the boosting variants. AdaBoost and RealBoost overemphasizes on the atypical examples which eventually

TABLE 3
OVERALL COMPARISON ON AP, BEP AND MRR (%) FOR VISUAL CONCEPT LEARNING

(a) Social Tags

	AP	BEP	MRR
Uniform	**74.8**	69.8	64.2
Distance	72.1	68.8	70.3
Rank	74.6	**70.0**	70.7
Ada	71.3	67.8	70.1
Real	67.5	64.8	55.9
Gentle	71.9	68.5	71.0
SVM	73.6	69.5	**74.2**

(b) Manual Tags

	AP	BEP	MRR
Uniform	82.8	77.2	68.0
Distance	83.6	77.8	72.3
Rank	84.3	78.3	72.8
Ada	84.6	78.6	61.9
Real	81.3	75.8	64.5
Gentle	85.0	78.8	72.6
SVM	**86.9**	**80.0**	**78.9**

*The best performance among all methods in terms of each evaluation criterion is underlined and **bolded**.*

TABLE 4
COMPARISON ON AP, BEP AND MRR (%) FOR IMPORTANCE WEIGHTED CONCEPT LEARNING

	AdaC2(i)	AdaC3(i)	Gentle-IW(i)	SVM-IW(i)	AdaC2(c)	AdaC3(c)	Gentle-IW(c)	SVM-IW(c)
AP	73.0	72.1	72.0	75.8	73.6	73.1	72.1	**76.1**
BEP	68.1	67.5	68.5	**71.3**	69.2	68.6	68.7	71.2
MRR	66.5	60.0	72.1	75.0	63.4	61.1	71.5	**75.1**

*Gentle-IW and SVM-IW denote the importance weighted GentleBoost and SVM. (i) and (c) denote per-image and per-concept weighing scheme for tag relevance-based importance weighting. The better performance between methods using each weighing scheme is underlined, while the best performance among all methods is **bolded**.*

results in inferior rules. By contrast, GentleBoost is numerically robust, and gives less emphasis to misclassified examples at each round since the increase in the weight of the example is quadratic in the negative margin, rather than exponential [57]. Additionally, combining the weak learners trained on multiple features at each round of boosting procedures consistently has a beneficial effect on all the boosting variants, since the uniform or error-based weighting scheme completely outperforms the feature selection approach (selecting the best one). In general, the uniform weighting works slightly better than the error-based weighting. However, the contrary is the case for GentleBoost when using social tags. Using manual annotations greatly improves the performance of using social tags. But the improvement is more noticeable in terms of AP and BEP than in MRR. In particular, the MRR score of AdaBoost even drops a little when using manual annotations. The reason for this might be that there are much less training examples in the manually labeled dataset. We note that the boosting algorithm might be improved, particularly in terms of MRR performance, using other multi-class algorithms, such as [58]. In the following experiments of visual concept learning, we just consider the better performing uniform weighting scheme in all the boosting algorithms for comparisons.

7.2.2 Learning Visual Concepts

In addition to boosting algorithms, we also use SVMs to learn separate classifiers for each concept by one-against-all strategy. In order to rank the concepts for a given image we need to compare the output scores of different SVM classifiers. To this end we perform cross-validation on the training data to fit a sigmoid function to map the SVM scores to probabilities. The regularization parameter C of the SVMs is also optimally selected by 5-fold cross-validation.

In Table 3 we present the overall results of all the visual concept learning algorithms described in this paper. As illustrated in Table 3(a) when learned from social tagged images, the visual neighbor voting model using uniform weights and rank-based weights obtain the best results in terms of AP and BEP respectively, while the SVM approach outperforms other classification algorithms in terms of concept ranking evaluation. In Table 3(b), by contrast, we observe an obvious improvement in performance when trained using manual annotations. SVMs now achieve the best performance in terms of all of our evaluation criteria. Furthermore, in both cases, visual neighbor voting model using rank-based weights and GentleBoost classifier give more competitive performance than other variants of the tag relevance learning model or the boosting algorithm. We have to emphasize that SVM classifier exhibits more powerful discriminative capabilities between semantic concepts than all the other classifiers in our experiments, as it yields much higher MRR scores in both cases.

In order to feed the importance weights to our importance weighted classifiers, we first perform tag relevance learning on the training dataset. Specifically, we learn the tag relevance of each training example by visual neighbor voting in a leave-one-out manner. Here the unsupervised tag relevance learning model using uniform weights is preferred, since the supervised learning models require manually labeled training data. We also study two relevance-based importance weighting schemes, i.e., per-image and per-concept weighting, to convert the tag relevance into importance weights for each training example. Apart from this, we use the same configurations, such as the choice of kernel function in SVM or weak learner in boosting, as above for our importance weighted SVMs and boosting algorithms in the following experiments.

As shown in Table 3(a) and Table 4, the cost-sensitive extensions of AdaBoost, i.e., AdaC2 and AdaC3, have very poor performance in terms of MRR, while they make some improvements in AP or BEP in comparison to classic AdaBoost without using importance weights. The importance weighted GentleBoost works much better than them. In particular, compared with original GentleBoost, its MRR score increases by up to 1.1%, which is hard to achieve for GentleBoost even by using manual annotations. Incorporating the importance weights into SVM classifiers gives the best performance. And the largest improvement made in terms of AP, BEP and MRR score is 2.5%, 1.7% and 0.9% respectively. An obvious flaw of our importance weighted classification is that for visual concepts that have large intra-class variations, it may fail to learn the example images with relatively rare visual appearance, since these examples probably have less visual neighbors in the training dataset, thus have smaller importance weights. As a result, the semantic concepts that are hard to learn due to intra-class variations will become harder to learn in our methods.

Table 5 lists the performance in terms of AP for all 20 annotation concepts in our evaluation dataset. It reveals that only around 52% of the user-supplied annotation concepts are truly related to the visual content of the training images. In general, concepts with higher user tagging accuracy achieve higher AP scores. For example, the most precisely user-labeled concept "flower" yields much higher score than the others when training with social tags, and the concept "lion" obtains a significant improvement when using manual annotations. However, some concepts can still perform well even with bad tagging accuracy, such as "kitchen" and "classroom". On the other hand, there is no obvious rise in terms of AP score for semantic concepts, such as "boat", even though when learning from manually annotated images. And similar observations can be made on the performance in terms of BEP and MRR which are not given here.

8 Conclusion

We have first explored two traditional classification algorithms, i.e. SVM and boosting variants, for visual concept learning using multiple image features. Then a visual neighbor voting model was presented to learn the relevance of tags with respect to the image content. In our experiments, we considered both the use of social tags and manual annotations of training images to evaluate the proposed methods. The results show that visual neighbor voting model works well for image ranking when learning from user-tagged images, while SVM classifiers perform best using manual annotations. As for concept ranking, the SVM approach exhibits better discriminative capabilities in both cases. Visual neighbor voting model using rank-based weights and GentleBoost classifier also achieve more comparable performance than other variants of the tag relevance learning model or the boosting algorithm. Additionally, using social tags is largely outperformed by using manual tags, since user-contributed tags tend to be subjective and noisy.

Indeed, for a given concept, relevant images have to be emphasized more in the training process than irrelevant images. Therefore we introduced an importance weighted extension to incorporate the example-dependent importance weights into SVM and boosting classifiers. These importance weights are acquired by exploiting the tag relevance with respect to each training image. And then our importance weighted classification is based on cost-sensitive learning, since high importance bears a high misclassification cost. Experimental results demonstrate that our methods, in particular SVMs, make obvious improvements, in comparison to learning without using importance weights. An empirical study on the impact of user tagging towards concept learning exhibits that, in general, better tagging accuracy leads to better performance, although some semantic concepts remain hard to learn in our experiments.

In future work we want to estimate the importance weights using supervised tag relevance learning models or using manually annotated images for neighbor voting which is expected to improve the performance of our importance weighted classification methods. Furthermore, designing more effective cost-sensitive extensions of SVM and GentleBoost classifiers is also an interesting and promising topic for us.

Acknowledgment

I would like to express my gratitude to Dr. Michael S. Lew who provides me with many valuable comments and suggestions in this project.

References

[1] M.S. Lew, N. Sebe, C. Djeraba, and R. Jain, "Content-based Multimedia Information Retrieval: State of the Art and Challenges," *ACM Trans. Multimedia Computing, Communications, and Applications*, vol. 2, no. 1, pp. 1-19, Feb 2006, doi: 10.1145/1126004.1126005.

[2] C. Wang, L. Zhang, and H.J. Zhang, "Learning to Reduce the Semantic Gap in Web Image Retrieval and Annotation," *Proc. ACM SIGIR Research and Development in Information Retrieval (SIGIR '08)*, pp. 355-362, 2008.

[3] R. Datta, D. Joshi, J. Li and J.Z. Wang, "Image Retrieval: Ideas, Influences, and Trends of the New Age," *ACM Comput. Surv.*, vol. 40, no. 2, pp. 1-60, May 2008, doi: 10.1145/1348246.1348248.

[4] Flickr Blog, "5 Billion Photos on Flickr," http://blog.flickr.net /2010/09/19/5000000000/. 2010.

TABLE 5
COMPARISON ON AP (%) FOR ALL 20 CONCEPTS

(a) Social Tags

AP	Uniform	Distance	Rank	Gentle	SVM	Gentle-IW	SVM-IW	Tagging Accuracy
airplane	49.4	57.3	52.2	50.0	53.6	51.0	51.2	45.3
beach	68.5	68.4	67.7	66.6	68.8	68.9	71.8	33.1
boat	57.2	58.6	59.4	55.6	53.2	55.2	58.8	44.9
bridge	85.7	86.1	86.7	86.1	86.3	87.0	86.9	76.6
bus	91.5	90.4	92.0	92.5	94.5	92.6	94.3	62.8
butterfly	92.7	85.0	88.2	86.5	91.3	86.4	93.1	68.8
car	82.6	82.5	83.2	78.7	82.0	79.3	83.1	55.2
cityscape	97.4	91.6	96.2	91.1	91.6	90.5	96.4	64.0
classroom	75.5	66.1	76.6	65.0	76.8	61.6	76.5	38.6
dog	87.1	83.8	85.4	88.3	88.9	88.6	88.6	75.2
flower	96.6	97.0	97.1	97.8	97.5	97.7	97.7	82.9
harbor	78.2	70.3	74.6	68.0	69.8	68.4	76.6	50.4
horse	86.2	87.5	89.3	82.7	83.1	83.1	85.8	73.6
kitchen	84.3	81.3	84.7	81.2	88.9	84.1	89.2	38.6
lion	48.2	45.0	46.1	45.1	39.4	45.2	48.3	34.6
mountain	82.7	80.0	83.6	83.1	83.7	84.0	85.5	47.6
rhino	70.4	61.3	73.2	60.7	71.8	62.6	75.5	36.0
sheep	75.3	64.3	68.3	74.0	70.1	72.8	75.1	53.0
street	69.5	69.1	71.0	68.2	66.1	66.3	71.6	43.8
tiger	16.7	16.6	16.5	16.5	15.3	16.8	16.6	23.4
Mean	74.8	72.1	74.6	71.9	73.6	72.1	**76.1**	52.4

(b) Manual Tags

AP	Uniform	Distance	Rank	Gentle	SVM
airplane	71.2	76.2	77.2	69.6	80.9
beach	70.2	69.6	70.9	73.4	75.1
boat	58.1	59.8	61.2	62.6	61.9
bridge	85.2	86.4	86.5	87.6	88.9
bus	91.8	92.4	92.6	93.9	95.5
butterfly	93.8	93.5	94.0	94.1	94.6
car	83.9	84.7	84.6	84.0	84.9
cityscape	98.1	97.8	98.4	98.0	97.3
classroom	81.9	79.0	80.9	83.0	86.5
dog	87.3	86.2	86.3	90.1	90.7
flower	96.3	96.5	96.6	97.2	97.3
harbor	90.0	90.4	90.1	91.6	92.2
horse	88.7	91.9	91.5	85.7	89.4
kitchen	85.3	84.5	85.5	88.1	91.3
lion	65.5	68.8	70.8	78.7	79.3
mountain	83.6	85.6	85.7	86.2	87.2
rhino	83.4	86.4	86.9	88.9	91.3
sheep	77.2	76.7	78.0	81.7	81.6
street	74.7	73.3	75.8	76.3	78.6
tiger	90.1	91.9	91.5	88.7	92.8
Mean	82.8	83.6	84.3	85.0	**86.9**

Comparison in terms of AP of all 20 concepts, as well as their mean. (a) and (b) illustrate the results when learning from social tags and manual tags respectively. Only the best performing boosting algorithm–GentleBoost, and its importance weighted extension Gentle-IW are given here. In addition, per-concept weighting is used for Gentle-IW and SVM-IW. The user tagging accuracy of each concept in our training dataset is also given at the last column in (a). The best performance among all methods for each concept is color filled and underlined, while the highest mean values are both underlined and ***bolded****.*

[5] G. Quenot, A. Tseng, B. Safadi, and S. Ayache, “TRECVID 2010 Collaborative Annotation,” http://mrim.imag.fr/tvca2010/. 2010.

[6] S. Richmond, “YouTube users uploading two days of video every minute,” *Daily Telegraph*, http://www.telegraph.co.uk /technology/goole/8536634/YouTube-users-uploading-two-days-of-video-every-minute.html/. Retrieved May 26, 2011.

[7] M.J. Huiskes, B. Thomee, and M.S. Lew, “New Trends and Ideas in Visual Concept Detection,” *Proc. ACM Int’l Conf. Multimedia Information Retrieval (MIR ’10)*, pp. 527-536, 2010.

[8] S.-F. Chang, J. He, Y.-G. Jiang, E.E. Khoury, C.-W Ngo, A. Yanagawa, and E. Zavesky, “Columbia University/VIREO-CityU/IRIT TRECVID 2008 High-level Feature Extraction and Interactive Video Search,” In *TRECVID 2008*, 2008.

[9] S. Zhu, G. Wang, C.-W. Ngo, and Y.-G. Jiang, “On the Sampling of Web Images for Learning Visual Concept Classifiers,” *Proc. ACM Int’l Conf. Image and Video Retrieval (CIVR ’10)*, pp. 50-57, 2010.

[10] A. Makadia, V. Pavlovic, and S. Kumar, “A New Baseline for Image Annotation,” *Proc. European Conf. Computer Vision (ECCV ’08)*, pp. 316-329, 2008.

[11] X. Li, C.G.M Snoek, and M. Worring, “Learning Social Tag Relevance by Neighbor Voting,” *IEEE Trans. Multimedia*, pp. 1310-1322, 2009.

[12] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid, “Tagprop: Discriminative Metric Learning in Nearest Neighbor Models for Image Auto-annotation,” *Proc. IEEE Int’l Conf. Computer Vision (ICCV ’09)*, pp. 309-316, 2009.

[13] S. Nowak and P. Dunker, “Overview of the CLEF2009 Large-scale Visual Concept Detection and Annotation Task,” In *CLEF working notes 2009*, 2009.

[14] M.J. Huiskes and M.S. Lew, “The MIR Flickr Retrieval Evaluation,” *Proc. ACM Int’l Conf. Multimedia Information Retrieval (MIR ’08)*, pp. 39-43, 2008.

[15] K.E.A Sande, T. Gevers, and A.W.M Smeulders, “The University of Amsterdam’s Concept Detection System at ImageCLEF 2009,” In *CLEF working notes 2009*, 2009.

[16] K. Mikolajczyk and C. Schmid, “An Affine Invariant Interest Point Detector,” *Proc. European Conf. Computer Vision (ECCV ’02)*, pp. 128-142, 2002.

[17] T. Tuytelaars and K. Mikolajczyk, “Local Invariant Feature Detectors: A Survey,” *Foundations and Trends in Computer Graphics and Visions*, vol. 3, no. 3, pp. 177-280, 2008.

[18] E. Nowak, F. Jurie, and B. Triggs, “Sampling Strategies for Bag-of-Features Image Classification,” *Proc. European Conf. Computer Vision (ECCV ’06)*, pp. 490-503, 2006.

[19] D. Lowe, “Distinctive Image Features from Scale-invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[20] H. Bay, A. Ess, T. Tuytelaars, and L.V. Gool, “Speed-up Robust Features,” *Computer Vision and Image Understanding*, vol. 110, pp. 346-359, 2008.

[21] J.R.R. Uijlings, A.W.M. Smeulders, and R.J.H. Scha, “Real-time Visual Concept Classification,” *IEEE Trans. Multimedia*, vol. 12, no. 7, pp. 665-681, 2010.

[22] F. Moosmann, E. Nowak, and F. Jurie, “Randomized Clustering Forests for Image Classification,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, pp. 1632-1646, 2008.

[23] J. Gemert, C. Veenman, A. Smeulders, and J. Geusebroek, “Visual Word Ambiguity,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, pp. 1271-1283, 2009.

[24] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories,” *Proc. IEEE Int’l Conf. Computer Vision and Pattern Recognition (CVPR ’06)*, vol. 2, pp. 2169-2178, 2006.

[25] J. Yuan, Y. Wu, and M. Yang, “Discovery of Collocation Patterns: from Visual Words to Visual Phrases,” *Proc. IEEE Int’l Conf. Computer Vision and Pattern Recognition (CVPR ’07)*, pp. 1-8, 2007.

[26] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid, “Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study,” *International Journal of Computer Vision*, vol. 73, no. 2, pp. 213-238, 2007.

[27] S. Maji, A. Berg, and J. Malik, “Classification using Intersection Kernel Support Vector Machines is Efficient,” *Proc. IEEE Int’l Conf. Computer Vision and Pattern Recognition (CVPR ’08)*, pp. 1-8, 2008.

[28] P. Viola and M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features,” *Proc. IEEE Int’l Conf. Computer Vision and Pattern Recognition (CVPR ’01)*, vol. 1, pp. 511-518, 2001.

[29] A. Torralba, K.P. Murphy, and W.T. Freeman, “Sharing Visual Features for Multiclass and Multiview Object Detection,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 5, pp. 854-869, 2007.

[30] J. Verbeek, M. Guillaumin, T. Mensink, and C. Schmid, “Image Annotation with TagProp on the MIRFLICKR set,” *Proc. ACM Int’l Conf. Multimedia Information Retrieval (MIR ’10)*, 2010.

[31] L.S. Kennedy, S.-F. Chang, and I.V. Kozintsev, “To Search or to Label: Predicting the Performance of Search-based Automatic Image Classifiers,” *Proc. ACM Int’l Conf. Multimedia Information Retrieval (MIR ’06)*, 2006.

[32] K. Bischoff, C.S. Firan, W. Nejdl, and R. Paiu, “Can All Tags Be Used for Search,” *Proc. ACM Int’l Conf. Information and Knowledge Management (CIKM ’08)*, 2008.

[33] D. Liu, X.S. Hua, L. Yang, M. Wang, and H.J. Zhang, “Tag Ranking,” *Proc. ACM Conf. World Wide Web (WWW ’09)*, 2009.

[34] A. Ulges, C. Schulze, D. Keysers, and T. Breuel, “Identifying Relevant Frames in Weakly Labeled Videos for Training Concept Detectors,” *Proc. ACM Int’l Conf. Image and Video Retrieval (CIVR ’08)*, pp. 9-16, 2008.

[35] P.Gehler and S. Nowozin, “On Feature Combination for Multiclass Object Classification,” *Proc. IEEE Int’l Conf. Computer Vision (ICCV ’09)*, pp. 221-228, 2009.

[36] Y. Freund and R. Schapire, “A Decision-theoretic Generalization of Online Learning and An Application to Boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119-139, 1997.

[37] J. Friedman, T. Hastie, and R. Tibshirani, “Additive Logistic Regression: A Statistical View of Boosting,” *The Annals of Statistics*, vol. 38, pp. 337-374, 2000.

[38] X.C. Yin, C.P. Liu, and Z. Han, “Feature Combination using Boosting,” *Pattern Recognition Letters*, vol. 26, no. 14, pp. 2195-2205, 2005.

[39] X. Li, C.G.M Snoek, and M. Worring, “Unsupervised Multi-Feature Tag Relevance Learning for Social Image Retrieval,” *Proc. ACM Int’l Conf. Image and Video Retrieval (CIVR ’10)*, pp. 10-17, 2010.

[40] P. Turney, “Types of Cost in Inductive Concept Learning,” *Proc. Int’l Conf. Machine Learning WorkShop Cost-sensitive Learning (ICML ’00)*, pp. 15-21, 2000.

[41] P. Domingos, “MetaCost: A General Method for Making Classifiers Cost-Sensitive,” *Proc. ACM SIGKDD*, pp. 155-164, 1999.

[42] C. Elkan, “The Foundations of Cost-Sensitive Learning,” *Proc. 17th Int’l Joint Conf. Artificial Intelligence*, pp. 973-978, 2001.

[43] Z.-H. Zhou and X.-Y. Liu, “On Multi-Class Cost-Sensitive Learning,” *Proc. 21st Nat’l Conf. Artificial Intelligence*, pp. 567-572, 2006.

[44] B. Zadrozny, J. Langford, and N. Abe, “Cost-sensitive Learning by Cost-proportionate Example Weighting,” *Proc. IEEE 3rd Int’l Conf. Data Mining*, pp. 435-442, 2003.

[45] N. Abe, B. Zadrozny, and J. Langford, “An Iterative Method for Multi-Class Cost-Sensitive Learning,” *Proc. ACM SIGKDD*, pp. 3-11, 2004.

[46] G. Karakoulas and J. Shawe-Taylor, “Optimizing Classifiers for Imbalanced Training Sets,” *Proc. Neural Information Processing Systems Workshop (NIPS ’99)*, pp. 253-259, 1999.

[47] U. Brefeld, P. Geibel, and F. Wysotzki, “Support Vector Machines with Example Dependent Costs,” *Proc. European Conf. Machine Learning (ECML ’03)*, pp. 23-34, 2003.

[48] F.R. Bach, D. Heckerman, and E. Horvitz, “Considering Cost Asymmetry in Learning Classifiers,” *The Journal of Machine Learning Research*, vol. 7, pp. 1713-1741, 2006.

[49] W. Fan, S. Stolfo, J. Zhang, and P. Chan, “AdaCost: Misclassification Cost-sensitive Boosting,” *Proc. 16th Int’l Conf. Machine Learning (ICML ’99)*, pp. 97-105, 1999.

[50] K.M. Ting, “A Comparative Study of Cost-sensitive Boosting Algorithms,” *Proc. 17th Int’l Conf. Machine Learning (ICML ’00)*, pp. 983-990, 2000.

[51] Y. Sun, A.K.C. Wong, and Y. Wang, “Parameter Inference of Cost-sensitive Boosting Algorithms,” *Proc. 4th Int’l Conf. Machine Learning and Data Mining in Pattern Recognition*, pp. 21-30, 2005.

[52] R. Schapire, “The Boosting Approach to Machine Learning: An Overview,” In *MSRI Workshop on Nonlinear Estimation and Classification*, 2001.

[53] X. Li, C. Snoek, and M. Worring, “Social20: A ground-truth set for tag-based social image retrieval,” http://staff.science.uva.nl /~xirong/index.php?n=Research.TagRelevanceLearning, 2009.

[54] J. Huang, S. Kumar, M. Mitra, W. Zhu, and R. Zabih, “Image Indexing using Color Correlograms,” *Proc. IEEE Int’l Conf. Computer Vision and Pattern Recognition (CVPR ’97)*, 1997.

[55] H. Yu, M. Li, H. Zhang, and J. Feng, “Color Texture Moment for Content-based Image Retrieval,” *Proc. IEEE Int’l Conf. Image Processing (ICIP ’02)*, pp. 929-932, 2002.

[56] A. Oliva and A. Torralba, “Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope,” *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145-175, 2001.

[57] Y. Freund and R.E. Schapire, “Discussion of the Paper ‘Additive Logistic Regression: A Statistical View of Boosting’,” In *The Annals of Statistics*, vol. 28, no.2 , pp. 391-393, 2000.

[58] V. Guruswami and A. Sahai, “Multiclass Learning, Boosting, and Error-correcting Codes,” *Proc. 12th Annual Conf. Computational Learning Theory*, pp. 145-155, 1999.