Chapter 6

Multimodal Classification

Classification can be defined as a decision-making problem which decides whether a new object belongs to a certain class or not. This decision making procedure is purely based on the previous observations on a sample, called training data. Such a sample consists of data points along with the class or category to which the points belong. When this decision making is confined to two classes or categories, it is known as binary classification and the corresponding classifier is known as a binary classifier. But, object classification, on the other hand, involves a number of different categories. Here, the problem transforms to a *multi-class* classification. The complexity of the problem is compounded if the decisive factors are of varied nature. Often under such circumstances, multimodal classification offers the best solution. If the object under consideration has, say, only two modalities– image and text, then image features and text features are treated as two different sources of

information. Both the sources can aid in classification if used judiciously. In that case, post-feature-extraction stage, the features need to be combined appropriately. In the paper by Guillaumin *et al.* [38], an image classifier is learned by using image features as well as the associated keywords. A Multiple Kernel Learning (MKL) classifier serves this purpose by accumulating the information from the image content and keywords. Here, the MKL is contrived on two basic classifiers, support vector machine (SVM) and least-squares regression (LSR). These classifiers have been shown to perform well in most cases and hence we have chosen SVM as the core building block of our multi-modal object classification.

From the existing literature, it is evident that the combination of multiple complementary features always strengthens the classification accuracy than any single feature. Eventually, text and image features can be considered to be complementary to each other as the textual descriptions are easier to process but they provide very little information about image content while the visual content of images can hardly be described in the text. Under such circumstances where both text and image features are used for object classification, MKL is a popular approach for feature combination. Like any other technique, MKL has its pitfalls as argued by Hou *et al.* [43]. In this work, the authors investigate two fundamental feature combination methods– average combination and weighted average combination for multimodal-classification. The experimental results of this work prove that the weighted average combination method outperforms MKL in both accuracy and efficiency. Feature combination entails computing features weights. Apart from feature combination,

feature weighting also plays a vital role in multimodal object classification. A judicious feature selection followed by an insightful feature weighting, *i.e.* weighting features according to their informativeness, can increase the classification efficiency.

6.1 Methodology

Before we proceed to explain our approach for optimal weight assignment to features, let us take a look at the baseline methods for weight assignment as described by Hou *et al.* [43].

6.1.1 Average combination

In this method of combining features all participating features are assigned equal weights. The combined kernel function for the SVM framework can be stated mathematically in the form of the Equation 6.1.

$$ker^{*}(i, i^{'}) = (\frac{1}{x}) \sum_{j=1}^{x} ker_{j}(i, i^{'}) \tag{6.1}$$

where x is the number of participating features in the task. Finally, this kernel function, $ker^*(i, i^{'})$, is used as a binary classifier with linear SVM.

6.1.2 Weighted average combination

Instead of assigning equal weights to each feature, in this method weights are assigned based on their discriminative power, stemming from the fact that feature with higher discriminative power should be assigned with higher weight for accurate classification [43]. So, discriminative power of each kernel has to be determined first. The steps to determine the weights for kernels, in brief, are as follows:

i. A binary classifier is trained individually for each kernel.

ii. Let us say, a_c is the discriminative power obtained after training for each kernel c.

iii. Finally, each kernel will be assigned with the weight, $W_c = \frac{a_c^b}{\sum_{c=1}^{x} a_c^b}$, where $b = 0, 0.5, 1, 2, 3, ..., 40$

6.1.3 Hill Climbing for Weight Assignment

As stated in previous sections, average combination strategy disregards the relative importance of features. Weighted average combination tries to rectify this issue by assigning a discriminative power to individual kernels and then assign them weights accordingly. We posit that instead of an empirical analysis to determine optimal

weights, a learning based approach can provide better results. Of all the existing optimization algorithms, gradient descent based approaches are quite popular as they tend to be simpler to implement and understand. Gradient-based optimizers are Hill Climbing algorithms and therefore local optimization techniques. Although many sophisticated algorithms exist, they all depend on a suitable starting point. In actual practice, finding this starting point or seed point has been perceived to be a major hurdle when trying to do unsupervised, automatic optimizations. Typically finding such a point and shortening the parameter intervals so that the goal function can actually be evaluated requires several tries. Our intuition behind considering Hill Climbing as an optimization procedure stems from the fact that when fewer variables are involved the kernel functions are easier to design and evaluate and performs decently.

To assign weights through optimization, we keep sum of all the weights assigned to the features constant. The *sum* of the weights is a whole number such that all feature weight add up to. Our objective is to find the best possible combination of weights that can maximize our classification accuracy. So in this case, an initial seed point for Hill Climbing, also called weight tuple t, is computed by assigning each feature equal (or nearly equal) weights, w_i, such that $\sum_i w_i = sum$. Let us consider that total number of features present is N. So, initially, the seed weight for each feature will be given $w_c = \frac{sum}{N}$. Some random $sum\%N$ kernels will be given weight $1 + \frac{sum}{N}$, to adhere to the constraint of constant sum.

Once the initial seed has been determined, by the logic of Hill Climbing, we aim to optimize the first weight w_1 in the tuple. To do so, we increase (or decrease) w_1 by 6 units (since there are total seven features in our dataset, as explained later), while we correspondingly decrease (or increase) the weights $w_2, w_3, ..., w_7$ by 1 unit. In case, the feature weight w_1 needs to be increased, it is done through a procedure `steal_increase()`, while other feature weights are adjusted using `donate_decrease()` and vice-versa (indicated as Steps 2 and 3 in Algorithm 5). Now, with this seed or weights, we feed the features into the SVM classifier and find out the precision of the classification. If the result is not satisfactory, we continue updating w_1 until we reach its optimal value. This whole process is repeated for each of the weights $w_1, w_2, ..., w_7$ to finally produce final seed tuple $\bar{t}$. This whole procedure is presented as Algorithm 5.

6.1.4 Extended Hill Climbing for Weight Assignment

We are aware of the fact that constrained Hill Climbing frequently gets stuck on the local maxima. To overcome this issue, we propose an *Extended Hill Climbing* method for weight assignment. In this extended Hill Climbing method, we choose multiple seed points along each direction instead of a single seed point. If we plot the precision values of classifier against each feature, we get seven 2D plots, whose values denote the classifier accuracy of corresponding feature weights. When Hill Climbing is performed over these plots, we assume a default step size. Now, such a blind choice of intervals can be detrimental to our cause since it is easy to miss a

Algorithm 5: Constrained Hill Climbing

Data: Seed tuple t
Result: Final tuple $\bar{t}$

for $1 \leq d \leq N$ **do**
  $t_{inc} \leftarrow \texttt{steal\_increase}(t, d)$;
  $t_{dec} \leftarrow \texttt{donate\_decrease}(t, d)$;
  $t^r \leftarrow \texttt{classifier\_accuracy}(t)$;
  $t^r_{inc} \leftarrow \texttt{classifier\_accuracy}(t_{inc})$;
  $t^r_{dec} \leftarrow \texttt{classifier\_accuracy}(t_{dec})$;
  if $t^r_{inc} > t^r_{dec}$ **then**
    $dir \leftarrow 1$ ▷ Select direction to move in;
    $t_{curr} \leftarrow t_{inc}$;
    $t^r_{curr} \leftarrow t^r_{inc}$;
  end
  else
    $dir \leftarrow 0$;
    $t_{curr} \leftarrow t_{dec}$;
    $t^r_{curr} \leftarrow t^r_{dec}$;
  end
  while $t^r_{curr} > t^r$ **do**
    $\bar{t} \leftarrow t_{curr}$;
    $t_{res} \leftarrow t^r_{curr}$;
    if $dir = 1$ **then**
      $t_{curr} \leftarrow t_{inc}$;
      $t^r_{curr} \leftarrow t^r_{inc}$;
    end
    else
      $t_{curr} \leftarrow t_{dec}$;
      $t^r_{curr} \leftarrow t^r_{dec}$;
    end
  end
  return $\bar{t}$;
end

particular weight (peak) lying in between two interval points for which the classifier produces better accuracy than the current one. So, we assert that decreasing the size of interval or step size can lead to better classifier accuracy. Hence, instead of starting with a fixed seed tuple, we choose various seed tuples and repeatedly perform Hill Climbing over each of the seed points until the best results are obtained. For

Algorithm 6: Extended Hill Climbing

Data: Seed tuple t, Kernel weight sum sum, Interval k
Result: Final tuple $\bar{t}$

1 $\delta \leftarrow \lceil \frac{sum}{k} \rceil$;
2 $\bar{t} \leftarrow \phi$;
3 $\bar{t}^r \leftarrow 0$;
4 $\omega \leftarrow \mathbf{0}$;
5 **for** $1 \leq d \leq N$ **do**
6 **for** $1 \leq p \leq k$ **do**
7 $\omega_d \leftarrow \delta * p$;
8 $t \leftarrow dist\_rem(S - \omega_d, d)$;
9 $t \leftarrow$ `Constrained_Hill_Climbing()`;
10 $t^r \leftarrow$ `classifier_accuracy(t)`;
11 **if** $t^r > \bar{t}^r$ **then**
12 $\bar{t} \leftarrow t$;
13 $\bar{t}^r \leftarrow t^r$;
14 **end**
15 **end**
16 **end**
17 **return** $\bar{t}$;

determining convergence, we executed the experiment multiple times by varying these seed points. Let say k number of intervals are considered in each direction. Then the step size can be defined as $\delta = \lceil \frac{sum}{k} \rceil$. Extended Hill Climbing is described as Algorithm 6. We perform Hill Climbing with chosen seed points simultaneously over all the features (indicated as Step 8 in Algorithm 6).

6.2 Experimental Setup

Our experiment is executed on 20 classes of PASCAL VOC'07 dataset[1] which was collected for the PASCAL visual object classes challenge task. In this dataset, a total

[1] http://lear.inrialpes.fr/data/

Class	Trainval Images	Trainval Objects	Test Images	Test Objects
Aeroplane	238	306	204	285
Bicycle	243	353	239	337
Bird	330	486	282	459
Boat	181	290	172	263
Bottle	244	505	212	469
Bus	186	229	174	213
Car	713	1250	721	201
Cat	337	376	322	358
Chair	445	798	417	756
Cow	141	259	127	244
Dining Table	200	215	190	206
Dog	421	510	418	489
Horse	278	362	274	348
Motor Bike	245	339	222	325
Person	2008	4690	2007	4528
Potted Plant	245	514	224	480
Sheep	96	257	97	242
Sofa	229	298	223	239
Train	261	297	259	282
TV Monitor	256	324	229	308
Total	**5,011**	**12,608**	**4,952**	**12,032**

TABLE 6.1: Statistics of PASCAL VOC'07 dataset

of 9,963 annotated images are present. The data is divided into two main subsets: training/validation data (`trainval`), and test data (test), with the `trainval` data further divided into suggested training (`train`) and validation (`val`) sets. For each subset and class, the number of images (containing at least one object of the corresponding class) and number of object instances are shown in Table 6.1. Each class contains different numbers of training-test samples as depicted in Table 6.1. There are a total of 5011 images in training set and the rest 4592 are test set images. Available per class ground truth aids to judge class-wise classification efficiency. MATLAB codes for different image feature extraction are also provided with the

development kit. In our task we consider six image features (*i.e.* DenseHue, Gist, hvecs32,Rgb, HarrisshiftV3H1, LabV3H1) and one text feature which have been extracted by using above mentioned codes. The SVM classifier was borrowed from Vedaldi and Fulkerson [89]. For experimental purposes, we fixed the $sum = 100$.

All of our experiments were carried out on a 64-bit Intel(R) Core(TM) i7-3770 3.40 GHz processor equipped machine having 6GB RAM and running 64-bit Windows 8.1 Pro. We used MATLAB R2011a (Version 7.120.635) to run the codes. For evaluation we have considered average precision and Mean Average Precision (MAP) as metrics.

6.3 Results and Analysis

The experimental results are presented in Table 6.2. This table reflects the average precision computed over each class corresponding to the test set of images. MAP values are presented in the last row of the table. From, the figures depicted, it is clear that Extended Hill Climbing is the best performing approach among all. When per class performance is considered, except four classes (Bus, Car, Dining table, Horse), extended version of Hill Climbing method outperforms all other methods. Even among those four classes, Hill Climbing performs the best for "Dining table". For "Car" and "Horse", the difference between precision of weighted combination and extended Hill Climbing is at best minuscule. "Bus" class is an exception where weighted average supersedes all other methods by a significant margin. For best

Class	**Avg. combination**	**Weighted avg. combination**	**Hill Climbing**	**Extended Hill Climbing**
Aeroplane	0.6050	0.8376	0.7840	**0.8442**
Bicycle	0.3090	0.5109	0.4830	**0.5034**
Bird	0.5120	0.6379	0.6600	**0.7124**
Boat	0.4180	0.3899	0.4420	**0.5499**
Bottle	0.0600	0.1264	0.1470	**0.1763**
Bus	0.3460	**0.6113**	0.4930	0.3947
Car	0.4960	**0.6266**	0.5430	0.6072
Cat	0.5970	0.3376	0.2420	**0.6926**
Chair	0.2790	0.3468	0.2910	**0.3541**
Cow	0.1850	0.3376	0.2420	**0.4987**
Dining Table	0.2330	0.2854	**0.2909**	0.2904
Dog	0.5660	0.6135	0.5980	**0.6324**
Horse	0.5770	**0.7586**	0.6520	0.7524
Motor Bike	0.3700	0.4868	0.4230	**0.5937**
Person	0.7390	0.7670	0.7530	**0.7678**
Potted Plant	0.0590	0.2096	0.2310	**0.2897**
Sheep	0.2310	0.3679	0.3370	**0.5623**
Sofa	0.197	0.2096	0.2310	**0.2427**
Train	0.6700	0.7694	0.7420	**0.7810**
TV Monitor	0.2200	0.2175	0.2870	**0.3098**
MAP	0.3835	0.4960	0.4586	**0.5275**

TABLE 6.2: Per class comparison of various combination approaches

performing classes such as "Sheep", extended Hill Climbing exceeds weighted average combination by 52.84% and average combination method by 143.41%.

The inability to exceed other approaches in those four classes could be attributed to the fact that the step sizes that we considered for extended Hill Climbing were whole numbers and hence coarse. We plan to carry out a detailed investigation with finer step values as part of our future work.

6.4 Discussion

Multimodal object classification involves identifying relevant features and then finding an appropriate combination of them to achieve the best result. Often the features have to be weighted based on their importance or discriminatory power towards classification. Existing state-of-the-art proposes average combination and weighted average combination of kernels to achieve this. We argue that an intelligent optimization method such as Hill Climbing can fetch us better results. Since gradient descent approaches like Hill Climbing have an inherent shortfall, we propose a modified version of the same named Extended Hill Climbing to ameliorate the same. Using SVM as a classifier, we show through our experiments on a publicly available standard dataset that our proposed approach significantly outperforms the other existing combination methods in image classification task.