Learning relevant eye movement feature spaces across users

Zakria Hussain[*] University College London

Kitsuchart Pasupa[†] University of Southampton

John Shawe-Taylor[‡] University College London

Abstract

In this paper we predict the relevance of images based on a low-dimensional feature space found using several users' eye movements. Each user is given an image-based search task, during which their eye movements are extracted using a Tobii eye-tracker. The users also provide us with explicit feedback regarding the relevance of images. We demonstrate that by using a greedy Nyström algorithm on the eye movement features of different users, we can find a suitable low-dimensional feature space for learning. We validate the suitability of this feature space by projecting the eye movement features of a new user into this space, training an online learning algorithm using these features, and showing that the number of mistakes (regret over time) made in predicting relevant images is lower than when using the original eye movement features. We also plot Recall-Precision and ROC curves, and use a sign test to verify the statistical significance of our results.

CR Categories: G.3 [Probability and Statistics]: Nonparametric statistics, Time series analysis—; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Relevance feedback

Keywords: Feature selection, Eye movement features, Online learning, Nyström method, Tobii eye-tracker

1 Introduction

In this study we devise a novel strategy of generalising eye movements of users, based on their viewing habits over images. We analyse data from a search task which requires a user to view images on a computer screen whilst their eye movements are recorded. The search task is defined as finding images most related to bicycles, horses or transport and the user is required to indicate which images on each page are relevant to the task. Using these eye movement features we run an online learning algorithm [Cesa-Bianchi and Lugosi 2006], such as the perceptron, to see how well we can predict the relevant images. However, is it possible to improve the results of the perceptron when taking into account the behaviour of other users' eye movements? We present some preliminary results that answer this question in the affirmative.

The initial goal of this work was to use the Multi-Task Feature Learning (MTFL) algorithm of [Argyriou et al. 2008]. However, in order to use nonlinear kernels, the MTFL algorithm requires a low-dimensional mapping using a Gram-Scmidt, Eigen-decomposition, etc., in order to construct explicit features [Shawe-Taylor and Cristianini 2004]. However, simply using these low-dimensional mappings, without applying the MTFL algorithm also generates a common feature space. Instead of the mappings mentioned above, we chose to employ the popular Nyström approximation [Williams and Seeger 2000] using a greedy algorithm called sparse kernel principal components analysis [Hussain and Shawe-Taylor 2009]. After this low-dimensional representation we can project all new users into this space and run the perceptron algorithm. We show that learning in this new refined feature space allows us to predict relevant images early on in the search task, indicating that our low-dimensional feature space has helped construct a space which generalises eye movements across several different users.

The paper is set out as follows. In Sub-Sections 1.1 and 1.2 we describe the background techniques of Online learning and low-dimensional mappings. Section 2 describes our method in detail and provides a novel algorithm to construct low-dimensional features across users. We describe the experimental setup and results in Section 3 and conclude the paper in Section 4.

1.1 Background I: Online learning

We are given a sequence of examples (inputs) $\mathbf{x} \in \mathbb{R}^n$ in n-dimensional space. These examples will be the different eye movement features extracted from the eye tracker. Furthermore, each user indicates relevance of images (*i.e.*, outputs) $y \in \{-1, 1\}$ where 1 indicates relevance and -1 indicates irrelevance.[1] Therefore we will concentrate on the binary classification problem. We will also be given $T \in \mathbb{N}$ users, who produce an m-sample of input-output pairs $S_t = \{(\mathbf{x}_{t1}, y_{t1}), \ldots, (\mathbf{x}_{tm}, y_{tm})\}$ where $t \in T$. Finally, given any m-sample S we would like to construct a classifier $h_m : \mathbf{x} \mapsto \{-1, +1\}$ which maps inputs to outputs, where m indicates the number of examples used to construct the classifier h_m.

Given a classifier $h_\ell(\cdot)$ we can measure the regret incurred so far, by counting the number of times our classifier has made an incorrect prediction. More formally, given an $(\mathbf{x}, y)$ pair, we say a mistake is made if the prediction and true label disagree *i.e.*, $h_\ell(x) \neq y$. Therefore, given $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ and a classifier computed using ℓ inputs we have the regret:

$$\texttt{regret}(\ell) = \sum_{i=1}^{\ell} I(h_\ell(\mathbf{x}_i) \neq y_i) \tag{1}$$

where $I(a)$ is equal to 1 if a is true and 0 otherwise. The regret can be thought of as the cumulative number of mistakes made over time.

The protocol used for online learning (see [Cesa-Bianchi and Lugosi 2006] for an introduction) is described below in Figure 1. The

- Initialise $\texttt{regret}(0) = 0$
- Repeat for $i = 1, 2, \ldots$
    1. receive input $\mathbf{x}_i$ and construct hypothesis h_i
    2. predict output $h_i(\mathbf{x}_i)$
    3. receive real output y_i
    4. $\texttt{regret}(i) = \texttt{regret}(i-1) + I(h_i(\mathbf{x}_i) \neq y_i)$

Figure 1: *Online learning protocol*

idea is to minimise the regret over time. We would like to make the least number of mistakes throughout the online learning protocol. We will show in this paper that we can leverage useful information from other tasks to construct a low-dimensional feature space leading to our online learning algorithm having smaller regret than when applied using individual samples. We will use the Perceptron algorithm [Cesa-Bianchi and Lugosi 2006] in our experiments.

[*]Department of Computer Science: z.hussain@cs.ucl.ac.uk

[†]School of Electronics and Computer Science: kp2@ecs.soton.ac.uk

[‡]Department of Computer Science: jst@cs.ucl.ac.uk

[1]In practice all images not clicked as relevant are deemed irrelevant.

1.2 Background II: Low-dimensional mappings

The Gram-Schmidt orthogonalisation procedure finds an orthonormal basis for a matrix. The idea is commonly used in the machine learning community when nonlinear kernels are used, but when an explicit feature representation of the inputs is required [Shawe-Taylor and Cristianini 2004]. The Gram-Schmidt procedure can be used in this scenario to yield an explicit feature representation. However, Gram-Schmidt does not take into account the residual loss between the low-dimensional mapping and may not be able to find a rich enough feature space.

A related method, known as the Nyström method, has been proposed and recently received considerable attention in the machine learning community [Smola and Schölkopf 2000][Drineas and Mahoney 2005][Kumar et al. 2009]. As shown by [Hussain and Shawe-Taylor 2009] the greedy algorithm of [Smola and Schölkopf 2000] finds a low-dimensional space that minimises the residual error between the low-dimensional feature space and the original feature space. It corresponds to finding the maximum variance in the data and hence viewed as a sparse kernel principal components analysis. Due to these advantages, we will use the Nyström method in our experiments.

We will assume throughout that our examples $\mathbf{x}$ have already been mapped into a higher-dimensional feature space $\mathcal{F}$ using $\phi : \mathbb{R}^n \mapsto \mathcal{F}$. Therefore $\phi(\mathbf{x}) = \mathbf{x}$, but we will make no explicit reference to $\phi(\mathbf{x})$. Therefore, if our input matrix $\mathbf{X}$ contain features as column vectors then we can construct a kernel matrix $\mathbf{K} = \mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{m \times m}$ using inner-products, where $\top$ denotes the transpose. Furthermore, we define each kernel function as $\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$.

Let $\mathbf{K}_i = [\kappa(\mathbf{x}_1, \mathbf{x}_i), \ldots, \kappa(\mathbf{x}_m, \mathbf{x}_i)]^\top$ denote the ith column vector of the kernel matrix. Let $\mathbf{i} = (1, \ldots, k)$, $k \leq m$, be an index vector and let $\mathbf{K}_{\mathbf{ii}}$ indicate the square matrix constructed using the indices $\mathbf{i}$. The Nyström approximation $\tilde{\mathbf{K}}$, which approximates the kernel matrix $\mathbf{K}$, can be defined like so:

$$\tilde{\mathbf{K}} = \mathbf{K}_{\mathbf{i}} \mathbf{K}_{\mathbf{ii}}^{-1} \mathbf{K}_{\mathbf{i}}^\top .$$

Furthermore, to project into the space defined by the index vector $\mathbf{i}$ we follow the regime outlined by [Diethe et al. 2009], where by taking the Cholesky decomposition of $\mathbf{K}_{\mathbf{ii}}^{-1}$ to get an upper triangle matrix $\mathbf{R} = \operatorname{chol}(\mathbf{K}_{\mathbf{ii}}^{-1})$ where $\mathbf{R}^\top \mathbf{R} = \mathbf{K}_{\mathbf{ii}}^{-1}$, we can make a projection into this low-dimensional space. Therefore, given any input $\mathbf{x}_j$ we can create the corresponding *new* input $\mathbf{z}_j$ using the following projection:

$$\mathbf{z}_j = \mathbf{K}_{j\mathbf{i}} \mathbf{R}^\top . \quad (2)$$

We have now described all of the components needed to describe our algorithm. However, it should be noted that the projection of Equation (2) is usually only applied to a single sample S, whereas in our work we have T different samples $S_1, \ldots, S_T$. We follow a similar methodology described by [Argyriou et al. 2008] (who use Gram-Schmidt) using the Nyström method.

2 Our method

Given $T-1$ input-output samples $S_1, \ldots, S_{T-1}$ we have $\mathbf{X}_t = (\mathbf{x}_{t1}, \ldots, \mathbf{x}_{tm})^\top$ where $t = 1, \ldots, T-1$. We concatenate these feature vectors into a larger matrix $\hat{\mathbf{X}} = (\mathbf{X}_1, \ldots, \mathbf{X}_{T-1})$ and compute the corresponding kernel matrix $\hat{\mathbf{K}} = \hat{\mathbf{X}}^\top \hat{\mathbf{X}}$. This kernel matrix contains information from all $T-1$ samples and so we apply the sparse kernel principal components analysis algorithm [Hussain and Shawe-Taylor 2009] to find the index set $\hat{\mathbf{i}}$ (where $|\hat{\mathbf{i}}| = k < m$) needed to compute the projection of Equation (2). This equates to finding a space with maximum variance between all of the different user behaviours. Given a new users eye movement feature vectors $\mathbf{X}_T$ we can project it into this joint low-dimensional mapping and run any online learning algorithm in this space:

$$\mathbf{X}_T^\top \hat{\mathbf{X}} \hat{\mathbf{R}}^\top$$

where $\hat{\mathbf{R}} = \operatorname{chol}(\hat{\mathbf{K}}_{\hat{\mathbf{i}}\hat{\mathbf{i}}}^{-1})$. We choose an online learning algorithm due to the speed, simplicity and practicality of the approach. For instance, when presenting users with images in a Content-Based Image Retrieval (CBIR) system [Datta et al. 2008] we would like to retrieve relevant images as quickly as possible. Although we do not explicitly model a CBIR system, we maintain that a future development towards applying our techniques within CBIR systems could be useful in retrieving relevant images quickly. In the current paper we are more interested in the feasibility study of whether or not any important information can be gained by the eye movements of other users.

For practical purposes we will randomly choose a subset of inputs from each user in order to construct $\hat{\mathbf{X}}$ – in practice this should not effect the quality of the Nyström approximation [Smola and Schölkopf 2000]. However, for completeness we describe the algorithm without this randomness. The feature projection procedure is described below in Algorithm 1.

Algorithm 1: Nyström feature projection for multiple users

```
Input: training inputs $\mathbf{X}_1, \ldots, \mathbf{X}_{T-1}$, new inputs $\mathbf{X}_T$ and
    $k \leq m \in \mathbb{N}$
Output: new features $\mathbf{z}_{T1}, \ldots, \mathbf{z}_{Tm}$
1: concatenate all feature vectors to get
       $\hat{\mathbf{X}} = (\mathbf{X}_1, \ldots, \mathbf{X}_{T-1})$
2: Create a kernel matrix $\hat{\mathbf{K}} = \hat{\mathbf{X}}^\top \hat{\mathbf{X}}$
3: Run the sparse KPCA algorithm of [Hussain and
   Shawe-Taylor 2009] to find index set $\hat{\mathbf{i}}$ such that $k = |\hat{\mathbf{i}}|$
4: Compute:
       $\hat{\mathbf{R}} = \operatorname{chol}(\hat{\mathbf{K}}_{\hat{\mathbf{i}}\hat{\mathbf{i}}}^{-1})$
5: for $j = 1, \ldots, m$ do
6:     construct new feature vectors using inputs in $\mathbf{X}_T$
           $\mathbf{z}_{Tj} = \mathbf{x}_{Tj}^\top \hat{\mathbf{X}}_{\hat{\mathbf{i}}} \hat{\mathbf{R}}^\top,$    (3)
7: end for
```

Furthermore, based on these feature vectors we also train the Multi-Task Feature Learning algorithm of [Argyriou et al. 2008] on the $T-1$ samples by viewing each as a *task*. The algorithm produces a matrix $\mathbf{D} \in \mathbb{R}^{\delta \times \delta}$ where $\delta < m$, which after taking the Cholesky decomposition of $\mathbf{D}$ we have $\mathbf{R}_D = \operatorname{chol}(\mathbf{D})$, and hence a low-dimensional feature mapping $\mathbf{z}_{Tj} = \mathbf{z}_{Tj} \mathbf{R}_D$. Unlike our Algorithm 1 this low-dimensional mapping also utilises information about the outputs.

3 Experimental Setup

The database used for images was the PASCAL VOC 2007 challenge database [Everingham et al.] which contains over 9000 images, each of which contains at least 1 object from a possible list of 20 objects such as cars, trains, cows, people, *etc.* There were $T = 23$ users in the experiments, and participants were asked to view twenty pages, each of which contained fifteen images. Their eye movements were recorded by a Tobii X120 eye tracker [Tobii Technology] connected to a PC using a 19-inch monitor (resolution of 1280x1024). The eye tracker has approximately 0.5 degrees of accuracy with a sample rate of 120 Hz and used infrared lens to detect pupil centres and corneal reflection. Figure 2 shows an example image presented to a user along with the eye movements (in red).

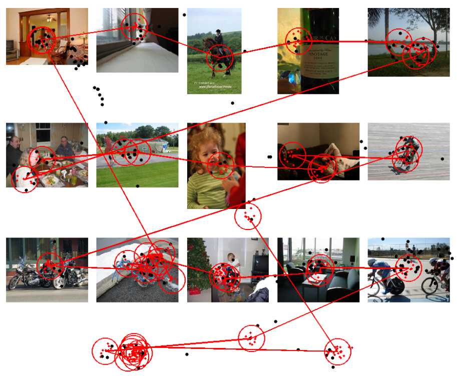


Figure 2: *An example page shown to a participant. Eye movements of the participant are shown in red. The red circles mark fixations and small red dots correspond to raw measurements that belong to those fixations. The black dots mark raw measurements that were not included in any of the fixations.*

Each participant was asked to carry out one search task among a possible of three, which included looking for a Bicycle (8 users), a Horse (7 users) or some form of Transport (8 users). Any images within a search page that a user thought contained images from there search, they marked as relevant.

The eye movement features extracted from the Tobii eye tracker can be found in Table 1 of [Pasupa et al. 2009]. Most of the features are motivated by features considered earlier for text retrieval studies, however in addition they also consider more image-based eye movement features such as the area of an image covered. The number of features extracted were 33. In the next subsection, these 33 features correspond to the "Original features".

Finally, in order to evaluate our methods with the perceptron we use the *Regret* (see Equation (1)), the *Recall-Precision (RP) curve* and the *Receiver Operating Characteristic (ROC) curve*.

3.1 Results

Given 23 users, we used the features of 22 users (*i.e.*, leave-one-user-out) in order to compute our low-dimensional mapping using the Nyström method outlined in Algorithm 1 – referred to as the "Nystrom features". We also made a mapping using the Multi-Task Feature Learning algorithm of [Argyriou et al. 2008], as described at the end of Section 2, which we refer to as the "Multi-Task features". The baseline used the Original features extracted using the Tobii eye tracker described above. The Original features did not compute any low-dimensional feature space and was simply used with the perceptron without using information from other users' eye movements.

Figure 3 plots the regret, RP and ROC curves when training the perceptron algorithm with the Original features, the Nystrom features and the Multi-Task features. We average each measure over the 23 leave-one-out users, and also over ten random splits of k feature vectors needed to construct $\hat{\mathbf{X}}$. It is clear that there is a small difference between the Multi-Task features and Original features in favour of the Multi-Task features. However, the Nyström features constructed using the Nyström method yield much superior results with respect to both of these techniques. The regret of the perceptron using the Nyström features is shown in Figure 3(a), and is always smaller than the Original and Multi-Task features (smaller regret is better). It is not surprising that we beat the Original features method, but our improvement over the Multi-Task features is more surprising as it also utilises the relevance feedback in order to construct a suitable low-dimensional feature space. However, the Nyström method only uses the eye movement features.

Figure 3(b) plots the Recall-Precision curve and shows dramatic improvements over the Original and Multi-Task features. This plot suggests that using the Nystrom features allows us to retrieve more relevant images at the start of a search. This would be important for a CBIR system (for instance). Finally, the ROC curves pictured in Figure 3(c) also suggest improved performance when using the Nyström method to construct low-dimensional features across users.

Algorithm	Average Regret	AAP	AUROC
Original features	0.2886	0.5687	0.6983
Nystrom features	**0.2433**	**0.6540**	**0.7662**
Multi-Task features	0.2752	0.5802	0.7107

Table 1: *Summary statistics for curves in Figure 3. The bold font indicates statistical significance at the level* $p < 0.01$ *over the baseline.*

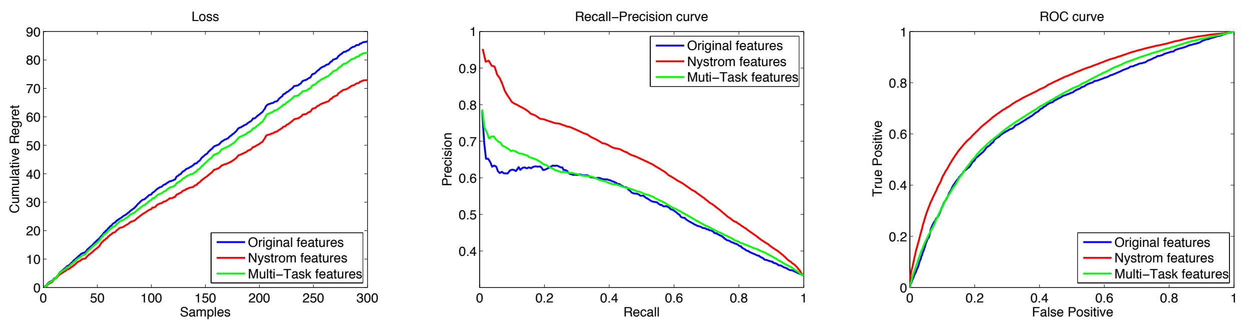
In Table 1 we state summary statistics related to the curves plotted in Figure 3. As mentioned earlier we randomly select k features from each data matrix $\mathbf{X}_t$ in order to construct $\hat{\mathbf{X}}$, and repeat this for ten random seeds – the results in Table 1 average over these ten splits. Therefore, for Figure 3(a), 3(b) and 3(c) we have the Average Regret, the Average Average Precision (AP), and the Area Under the ROC curve (AUROC), respectively. Using Nyström features are clearly better than using the Original and Multi-Task features with a significance level of $p < 0.01$. Though using Multi-Task features is slightly better than using the Original features, however, the results are only significant at the level $p < 0.05$. The significance level of the direction of differences between the two measures was tested using the sign test [Siegel and Castellian 1988].

4 Conclusions

We took eye gaze data of several users viewing images for a search task, and constructed a novel method for producing a good low-dimensional space for eye movement features relevant to the search task. We employed the Nyström method and showed it to produce better results than the extracted eye movement features [Pasupa et al. 2009] and a state-of-the-art Multi-Task Feature learning algorithm.

The results of this paper indicate that if we would like to predict the relevance of images using the eye movements of a user, then it is possible to improve these results by learning relevant information from the eye movement behaviour of other users who are viewing similar (not necessarily the same) images *and* given similar (not necessarily the same) search tasks.

The first obvious application of our work is to CBIR systems. By recording the eye movement of users we could apply Algorithm 1 to project such gaze features into a common feature space, and then



(a) The Regret (b) The Recall-Precision curve (c) The ROC curve

Figure 3: *Averaged over 23 users: results of the perceptron algorithm using Original, Nyström and Multi-Task features.*

apply a CBIR system in this space. Based on the results we reported within this paper, we would hope that relevant images were retrieved early on in the search. Another application to CBIR, would be to use the method we presented as a form of verification of relevance. Some users may give incorrect relevance feedback due to tiredness, lack of interest, *etc*., but based on our method we could correct such actions as and when users make judgements on relevance. As the perceptron would improve classification over time and we would expect users to make mistakes near the end of a search, then such corrective action would be useful to CBIR systems requiring accurate Relevance Feedback mechanisms to improve search.

The machine learning view is that we are projecting the feature vectors into a common Nyström space (constructed from feature vectors of different users), and then running the primal perceptron (not the kernel perceptron) with these features. Our experiments suggest better performance than simply using the perceptron together with the original features. Therefore, we would like to construct regret bounds for the perceptron when using this common low-dimensional space. Finally, for practical purposes we believe that randomly selecting columns from the kernel matrix to find $\hat{\mathbf{i}}$ would generate a feature space [Kumar et al. 2009] with little deterioration in test accuracy but increased computational speed-ups, and hence a direction for future work.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under *grant agreement* n° 216529, Personal Information Navigator Adapting Through Viewing, PinView.

References

ARGYRIOU, A., EVGENIOU, T., AND PONTIL, M. 2008. Convex multi-task feature learning. *Machine Learning 73*, 3, 243–272.

CESA-BIANCHI, N., AND LUGOSI, G. 2006. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA.

DATTA, R., JOSHI, D., LI, J., AND WANG, J. Z. 2008. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys 40*, 5:1–5:60.

DIETHE, T., HUSSAIN, Z., HARDOON, D. R., AND SHAWE-TAYLOR, J. 2009. Matching pursuit kernel fisher discriminant analysis. In *Proceedings of 12th International Conference on Artificial Intelligence and Statistics, AISTATS*, vol. 5 of *JMLR: W&CP 5*.

DRINEAS, P., AND MAHONEY, M. W. 2005. On the Nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research 6*, 2153–2175.

EVERINGHAM, M., VAN GOOL, L., WILLIAMS, C. K. I., WINN, J., AND ZISSERMAN, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

HUSSAIN, Z., AND SHAWE-TAYLOR, J. 2009. Theory of matching pursuit. In *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. 721–728.

KUMAR, S., MOHRI, M., AND TALWALKAR, A. 2009. Sample techniques for the Nyström method. In *Proceedings of 12th International Conference on Artificial Intelligence and Statistics, AISTATS*, vol. 5 of *JMLR: W&CP 5*.

PASUPA, K., SAUNDERS, C., SZEDMAK, S., KLAMI, A., KASKI, S., AND GUNN, S. 2009. Learning to rank images from eye movements. In *HCI '09: Proceeding of the IEEE 12th International Conference on Computer Vision (ICCV'09) Workshops on Human-Computer Interaction*, 2009–2016.

SHAWE-TAYLOR, J., AND CRISTIANINI, N. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, U.K.

SIEGEL, S., AND CASTELLIAN, N. J. 1988. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill, Singapore.

SMOLA, A., AND SCHÖLKOPF, B. 2000. Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th International Conference on Machine Learning*, 911–918.

TOBII TECHNOLOGY, L. Tobii Studio Help. http://studiohelp.tobii.com/StudioHelp_1.2/.

WILLIAMS, C. K. I., AND SEEGER, M. 2000. Using the Nyström method to speed up kernel machines. In *NIPS*.