Composite Discriminant Factor Analysis

Ejaz Ahmed
http://www.umiacs.umd.edu/~ejaz
The University of Maryland
College Park, USA

Abstract

We propose a linear dimensionality reduction method, Composite Discriminant Factor (CDF) analysis, which searches for a discriminative but compact feature subspace that can be used as input to classifiers that suffer from problems such as multi-collinearity or the curse of dimensionality. The subspace selected by CDF maximizes the performance of the entire classification pipeline, and is chosen from a set of candidate subspaces that are each discriminative by various local measures, such as covariance between input features and output labels or the margin between positive and negative samples. Our method is based on Partial Least Squares (PLS) analysis, and can be viewed as a generalization of the PLS1 algorithm, designed to increase discrimination in classification tasks. While our experiments focus on improvements to object detection (in particular, pedestrians and vehicles), a task that often involves high dimensional features and benefits from fast linear approaches, we also demonstrate our approach on UCF50 action recognition dataset and machine learning datasets from the UCI Machine Learning repository. Experimental results show that the proposed approach improves significantly over SVM in terms of accuracy and also over PLS in terms of compactness and efficiency, while maintaining or slightly improving accuracy.

1 Introduction

Dimensionality reduction methods have been popular in the computer vision community [9] as preprocessing tools to deal with the increasing dimensionality of input features. The literature includes linear methods [5, 10, 18]; non-linear methods, some of which are kernelized versions of linear methods [3, 6, 23, 25]; and feature selection methods [9]. We focus on linear feature construction methods that obtain compact but predictive features by linear transformations, motivated by the task of object detection, which involves high-dimensional features constructed from dense feature grids (e.g., HOG [7, 8], pyramidal HOG [33], dense SIFT [16]) and a detection step that repeatedly applies classifiers to features constructed from image sub-windows at varying scales, translations, and rotations. This sliding window detection process can benefit from linear projections in various ways. For instance, new samples are efficiently projected into the subspace by matrix multiplication and the high-dimensional training data does not need to be stored as it is for kernel methods, reducing memory and computational requirements. Additionally, linear projection can be performed efficiently by first extracting a feature grid for the entire image and then performing linear convolution [8]. This process avoids redundant feature computation for features that are included in multiple windows at different locations within each window, and also reduces memory copy operations since the projection is computed in place. Not surprisingly, a large number of state-of-the-art approaches use linear classifiers, typically Linear SVM [6].
Motivated by these trends, we propose a new approach that selects one or more linear projection vectors to produce a compact and discriminative subspace, optionally followed by a non-linear classification step (which is computationally cheap on low-dimensional inputs). This process is based on Partial Least Squares (PLS) \cite{21, 30}, a class of methods which model the relationship between two or more sets of observed variables via a set of latent variables chosen to maximize the covariance between the sets of observed variables. More specifically, our approach is based on the most frequently used variants of PLS \cite{21}, PLS1 and PLS2, both of which are used for regression by a process that iteratively obtains a projection vector that maximizes covariance between the input and response variables. Instead of using PLS directly, as has been done previously \cite{13, 26}, we use PLS internally to generate compact subspaces that improve the performance of our entire classification pipeline.

Our approach is based on the observations that 1) maximizing covariance between the input features and response variables does not necessarily yield a compact feature space for the purpose of classification, and 2) linear combinations of PLS factors obtained by performing regression from the latent space to the response variables are much more compact and almost as discriminative as the factors themselves. For binary classification, the composite is a projection vector. By varying how many factors are used to create a composite, we create a number of candidate projection vectors. Taking advantage of the deflation operation of PLS, we can iteratively alternate between selecting a composite direction and deflation to obtain multiple projection vectors that define a multidimensional latent subspace. The number of PLS factors that we consider for each composite and the number of deflation iterations that we perform together parametrize a set of candidate subspaces. From this set, we select the subspace that maximizes the performance of the entire classification pipeline, using cross-validation and best-first search.

One appealing property of our approach is that the proposed set of subspaces includes the one obtained by PLS, and so our approach can be viewed as a generalization of PLS. In addition, subject to mild constraints, other approaches can be used to propose projection vectors at each iteration. We show empirically that our process not only outperforms PLS and other state-of-the-art baseline approaches on a number of datasets, but it does so with only one or two-dimensional subspaces. We demonstrate the performance of our approach on the tasks of pedestrian detection on the INRIA Pedestrian dataset \cite{7} and vehicle detection in aerial images (45° and 90°) that we will make publicly available. In addition, we demonstrate our approach on four public datasets from the UCI Machine Learning repository \cite{2} and on the UCF50 \cite{1} action recognition dataset.

1.1 Related work

Linear methods have been used in the field of computer vision for dimensionality reduction or directly for classification. For example, Principal Component Analysis has been used as a dimensionality reduction approach for face recognition by \cite{27}, followed by Linear Discriminant Analysis (LDA) for face \cite{4}, pedestrian, and object recognition \cite{11}. Other methods, such as Canonical Correlation Analysis (CCA) have also been applied to vision \cite{14}.

A popular linear classifier and descriptor combination currently employed by a large number of state-of-the-art vision approaches is linear SVM \cite{5} and Histograms of Oriented Gradients (HOGs), initially applied by Dalal and Triggs \cite{7} to detect pedestrians. Subsequently, improved human detectors have been proposed that can handle partial occlusion \cite{29}. More general deformable part models (DPM) have been proposed that model objects
as a set of part filters anchored to a root filter that are applied to modified HOG features, and trained using an extension of linear SVM, called Latent SVM. Recently, Malisiewicz et al. train linear SVM classifiers on HOG descriptors of each in a one-vs-all fashion to every positive instance (or exemplar) available in the training set \cite{17}. Other approaches using these building blocks include: branch-and-bound detection applied to linear SVMs for efficient search \cite{15}; coarse-to-fine object localization \cite{20,33}; scale invariant detection at multiple resolutions, in which small instances are detected with rigid templates and large instances are detected by deformable part models \cite{19}; active learning\cite{28}, where a linear classifier is used to identify uncertain windows that need to be labeled manually; and pose-estimation \cite{32} using an approach similar to DPM. The success of the HOG and linear SVM combination may be due to a number of factors including: HOG robustly encodes edge information without requiring a thresholding step, and allows for slight deformations in edge locations by the use of small histograms; HOG can be computed as a grid after which linear projection can be performed efficiently by convolution; and SVM focuses on the classification margin, so it can handle extremely large negative sets by keeping in cache only hard negatives while maintaining convergence guarantees.

Other linear classifier approaches have been proposed as well. In particular, Partial Least Squares (PLS) \cite{30}, has been recently applied to the problem of human and vehicle detection \cite{13,26}, largely due to its ability to efficiently handle high dimensional data. Unlike PCA \cite{18}, PLS can be used as a class-aware dimension reduction tool, and unlike other class-aware dimension reduction tools, such as LDA\cite{10,18} or CCA \cite{10}, it can handle very high-dimensional data and its associated problems (multi-collinearity, in particular). While many PLS extensions exist such as Canonical PLS (CPLS) and Canonical Power PLS (CPPLS) \cite{12}, Kernel PLS \cite{22}, and others \cite{21}, we will focus on extensions to the standard linear PLS approach with the goal of improving existing linear approaches that are used in many of the vision systems described above. Our work is motivated by our observation that PLS often outperforms linear SVM—but that it requires a larger linear subspace (linear SVM can be seen as projecting into a single-dimensional subspace).

Our contribution consists of a new approach, CDF, which is based on PLS but yields more compact linear subspaces that can be used for training classifiers. The benefit of lower dimensional subspaces, provided that they preserve discriminability, is not just computational—more complex classification approaches often generalize better if presented with samples that lie in a lower dimensional subspace. In the following sections, we will briefly summarize PLS, introduce our proposed approach, and present experimental results on pedestrian detection, vehicle detection (Google 90° and Google 45°), action recognition and on benchmark machine learning datasets.

## 2 Partial Least Squares

A number of Partial Least Squares (PLS) variants model relations between two or more sets of observed variables through a set of latent variables; many of these are discussed in detail in \cite{21,30}. We briefly summarize the most frequently used variants, PLS1 and PLS2 \cite{21}, which relate two sets of observed variables $X \in \mathbb{R}^{n \times p}$ and $Y \in \mathbb{R}^{n \times q}$, and are generally used for regression problems. Here, $n$ is the number of observed samples, $p$ is the dimensionality of samples from $X$ and $q$ is the dimensionality of samples from $Y$. PLS1 is the special case where $q = 1$, while PLS2 is the more general case where $q > 1$. PLS decomposes the zero-mean matrices $X$ and $Y$ as follows:
\[ X = TP^T + E, \quad Y = UQ^T + F \]

where \( T \) and \( U \) are \( n \times f \) matrices containing \( f \) latent vectors \( t_i \) and \( u_i \) (the coefficients obtained by projecting into the latent space), \( P \in \mathbb{R}^{p \times f} \) and \( Q \in \mathbb{R}^{q \times f} \) contain the loadings, and \( E \in \mathbb{R}^{n \times p} \) and \( F \in \mathbb{R}^{n \times q} \) are the residuals that result from using only \( f \) latent vectors to reconstruct \( X \) and \( Y \). Usually the PLS decomposition is obtained by the nonlinear iterative partial least squares (NIPALS) algorithm [30], summarized in Algorithm 1, which iteratively constructs \( T \) and \( U \) one column at a time by finding the weight vectors \( w \) and \( c \) such that the covariance between the latent coefficients and some other classifier (e.g., QDA) is applied to the factors in Algorithm 1 via the power iteration loop on lines 2–8. Once weight vectors \( w \) and \( c \) are obtained, the normalized score vector \( t = Xw/||Xw|| \) is computed. The matrix \( X \) is deflated by its rank-one reconstruction from \( t \), and \( Y \) is deflated by the rank-one component of the regression of \( Y \) on \( t \) (Alg. 1, lines 9–10). The deflation step guarantees that subsequent weight vectors \( w_{i+1} \) and resulting score vectors \( t_{i+1} \) explain only the residuals, and thus are independent, i.e. \( T^T T = I \) and \( W^T W = I \), where \( t_i \) and \( w_i \) are the \( i \)th columns of \( T \) and \( W \). It can be shown that \( P = X^T T \) minimizes reconstruction error \( ||E||_2 \). Because the columns of \( W \) are computed from deflated data, we compute a matrix \( W_s = W(P^TW)^{-1} \) that corrects for the deflation step so that we can obtain the latent scores (or coefficients) of \( X \) by a linear projection, \( T = XW_s \).

**Figure 1:** Algorithm 1: PLS(NIPALS version)

1: for \( i = 1, \ldots, f \) do
2: \( u_i \leftarrow y_1/||y_1|| \)
3: repeat
4: \( w_i \leftarrow X^T u_i/||X^T u_i|| \)
5: \( t_i \leftarrow Xw_i/||Xw_i|| \)
6: \( c_i \leftarrow Y^T t_i/||Y^T t_i|| \)
7: \( u_i \leftarrow Yc_i \)
8: until convergence
9: \( X \leftarrow X - t_it_i^T X \)
10: \( Y \leftarrow Y - t_it_i^T Y \)
11: end for
end

**Figure 2:** Algorithm 2: CDF

1: for \( i = 1, \ldots, f \) do
2: \( w_i \leftarrow \text{pls_composite}(X, Y, n_i) \)
3: \( w_i \leftarrow w_i/||w_i|| \)
4: \( t_i \leftarrow Xw_i/||Xw_i|| \)
5: \( X \leftarrow X - t_it_i^T X \)
6: \( Y \leftarrow Y - t_it_i^T Y \)
7: end for
end

PLS classification can be performed by letting \( X \) be the input features and \( Y \) be the \( n \times c \) class indicator matrix for multiclass classification or a \( n \times 1 \) indicator vector for the binary case. If PLS is used for feature extraction, then \( f \) factors are extracted as linear combinations of the input features, and some other classifier (e.g., QDA) is applied to the factors \( T = XW_s \). Note that because \( T^TT = I \), the projected data is also whitened in the process, a
AHMED: COMPOSITE DISCRIMINANT FACTOR ANALYSIS

Figure 3: Motivating examples. **Left:** Example of how initial PLS dimensions are influenced by input feature covariance. A 3-dimensional dataset is generated by sampling from a Gaussian distribution with standard deviations of [0.5, 4, 1] on the diagonal, rotating by 45 degrees in the x-y plane, and shifting the class means apart. The plots show the projection of all points on the x-y plane. The first PLS factor is visibly influenced by the principal axis, causing confusion between the two classes when points are projected onto the factor. The second factor corrects for this, and the third reverses some of the correction. **Middle:** the composites of the factors on the left. In this toy example two factors are enough to create a discriminative composite (a single projection vector). **Right:** comparison between classification error obtained by QDA on $f$ PLS factors (an $f$-dimensional subspace) versus the composite of the first $f$ factors (a 1-dimensional subspace); trained and evaluated on the gisette training and validation subsets, respectively.

preprocessing step that often improves classifier performance. Alternatively, classification can be performed by linear regression, predicting the indicator matrix from the input features by $Y = XB + G$, where $B = W(P^T W)^{-1}P^TY = W T^T Y$ and $G$ is a residual matrix. The only parameter for PLS is the number of factors $f$ needed for regression or feature extraction, and is usually set by cross-validation.

## 3 Composite Discriminant Factors

While PLS has been successfully used to select subspaces that are discriminative for classification tasks, the factors that are chosen are not very compact. For example, in Figure 3 the initial factor is affected by the covariance of the data $X$, which in this case is not informative for discrimination. By extracting sufficient factors, PLS eventually overcomes this problem. The middle plot shows the composite projection vector, a single vector computed as a linear combination of the $f$ PLS factors (which is why we call it a composite) by PLS regression. It is evident that because PLS regression maps from the latent space to the class indicator, the composite is able to encode the discriminative direction in a single vector. The two plots on the left of Figure 3 are toy examples, but the pattern appears in real data as well—the third plot is only one of many examples where a single composite matches and even outperforms Quadratic Discriminant Analysis (QDA) applied to the $f$ factors from which the composite is computed. These examples suggest that while a large set of latent factors that maximize covariance may lead to good discrimination, it is possible to achieve the same results with a more compact set of factors, motivating our approach Composite Discriminant Factors (CDF).

Just as the PLS algorithm alternates between computing a factor and deflating the data matrices, we can iterate CDF as well, in this case between computing a composite and deflating by that composite. It is easy to show that as long as the composite is a linear combination of the rows of the deflated $X$, the properties of the PLS deflation process are satisfied, i.e.,
AHMED: COMPOSITE DISCRIMINANT FACTOR ANALYSIS

Figure 4: Visualization of CDF parameter space. Each level below the root corresponds to an additional composite. The highlighted path corresponds to the original PLS algorithm (thus CDF generalizes PLS in that sense) and should at least match PLS performance, provided sufficiently good model selection during training.

\[ W^T W = I \text{ and } T^T T = I. \] The composite \( B \) is in the row span of \( X \), since it is a linear combination of factors which are each in the row span of \( X \). CDF is parametrized by a list \( (n_1, n_2, \ldots, n_f) \) of length \( f \), containing the number of factors \( n_i \) to use for the \( i \)th composite, and proceeds in a similar fashion to PLS, as shown in Algorithm 2.

The parameter space is now much larger than that of PLS, each parameter list representing a linear subspace obtained from the row span of \( X \), and is depicted visually as a tree in Figure 4. The root node corresponds to the original input data \( X \), edges correspond to candidate composites, and child nodes correspond to parent nodes deflated by the composite along the edge. In Figure 4 we denote PLS and CDF, along with their parameters, by \( \text{pls}(f) \) and \( \text{cdf}(n_1, \ldots, n_f) \), respectively. It is easy to see that \( \text{cdf}(n_1 = 1, \ldots, n_f = 1) = \text{pls}(f) \), so PLS can be represented in the CDF parameter space. Because this parameter space is so large, we propose a best-first search algorithm for the CDF subspace that is optimal for a classification task, potentially with some bounded depth. The search process proceeds by opening children of the node that has so far yielded the best cross-validation score. Here, “opening” a node means that CDF with the corresponding parameters is instantiated and evaluated by cross-validation. Once the search terminates, the parameters corresponding to the node with the best cross-validation score are chosen.

Although CDF composites have so far been obtained by nested iterations of PLS, other projection directions can be considered as well. For example linear SVM weight vectors are linear combinations of support vectors, so they are also in the row span of \( X \). In this case, a copy of original uncentered \( Y \) indicator matrix is needed at line 2 of Algorithm 2, instead of the deflated \( Y \) matrix used for PLS. Other approaches, such as CPLS or CPPLS [31] could be used to propose projection directions. We will focus on CDF with composites obtained by PLS in this paper, leaving other methods for future work.

4 Experiments

UCI Machine Learning Repository. We evaluate the performance of CDF on four standard benchmark datasets from the NIPS 2003 Feature Selection Challenge [9]: arcene, dexter, dorothea, and gisette, available from the UCI Machine Learning Repository [2]. Table 1 shows our results on datasets. We select parameters for CDF, PLS, and SVM using 20-fold cross-validation, selecting the value of \( C \) from \( 10^{-7} \) to \( 10^7 \) in powers of 10, up to 20 factors for PLS, and up to 11 PLS factors per CDF composite. We use QDA as the non-linear classifier after PLS or CDF projection. We bound our CDF search at a depth of two (so we find at most two factors), since CDF already matches the performance of SVM and PLS.
with only one or two composites. Our classification results are relatively invariant to scaling for all but the *arcene* dataset, which we normalize by scaling each feature by its standard deviation (the relative performance between SVM, PLS, and CDF remains fixed when even when arcene is not scaled). A noteworthy result is that CDF achieved the reported error rates with 1 composite for *arcene* and *dexter* and 2 composites for *dorothea* and *gisette*. PLS required 6, 4, 6, and 17 factors for the four datasets (in order).

**Vehicle Detection: Google 90° Satellite Dataset.** We evaluated the performance of our system on satellite images taken from Google. The dataset consists of 24 high resolution satellite images (RGB) at a resolution of 2048 x 2048 pixels. Train and test sets contain 12 images each with 708 and 1054 vehicles respectively. Training is done by selecting negatives using hard negative mining. We sampled 100 negatives per training image at random locations and orientations giving us 1200 initial negatives and 708 positives for training the bootstrap model. We select up to 800 hard negatives in each retraining iteration. We use multi-scale HOG features similar to [33]. Features are calculated for blocks ranging from a size of 12 x 12 pixels to 30 x 66 pixels (window size). Each block yields a 36D feature vector resulting in a total length of 8424. For training, we consider up to 8 PLS factors for PLS and CDF training. Parameters are selected by 20-fold cross validation. Following the example of [26], we use QDA as the classifier after projection. We use horizontal and vertical step size of 1 pixel and we consider every 10 degrees of rotation. Approximately 144 million windows are processed per image. Our proposed approach outperforms linear SVM by a large margin and also outperforms PLS in the high recall region Figure 5. PLS training chose 8 factors during cross-validation, thus is significantly slower during test time. We also tested PLS by increasing the factors to 20 and still found that classifier with 8 factors was chosen during training. The 2 composites of CDF were obtained by obtaining the composite of first 6 PLS factors, deflating by this composite and then using 1 PLS factor from the deflated data. The weight vectors in Figure 5 show us that CDF and PLS capture finer details of the vehicle like windows of the car, while SVM just captures contour of the car. To asses the variation in performance due to the initial random negative set, we repeated the retraining process 10 times, and found that the average precision of the 10 resulting PR curves had a standard deviation of 0.00366.

**Vehicle Detection: Google 45° Satellite Dataset.** We also tested our method on a more challenging vehicle dataset of high resolution 45° oblique view Google satellite images. Because the images are captured at a 45° angle, there is a large amount of variation in vehicle appearance. The dataset contains 171 RGB images at a resolution of 1720 x 950. We randomly divided the images into a training set (85 images, 1602 vehicles) and a test set (86 images, 2006 vehicles), and trained the detector as described above. We used window size of 64 x 112 pixels along with Felzenszwalb HOG features with a bin and stride of 4 resulting in 9216 dimensional feature vector. Both PLS and CDF outperform SVM by a large margin Figure 5. During training, we considered up to 10 PLS factors, and chose 9 factors as optimal by cross-validation. CDF gives slightly better performance than PLS, especially at higher recall, but the most important result is that it does so with only 2 composites, making CDF significantly faster than PLS during test time Figure 7. We plan to make both vehicle
datasets publicly available.

**Pedestrian Detection: INRIA Pedestrian Dataset.** We also compare the performance of our classifier as part of a human detector on publicly available INRIA Pedestrian Dataset [7], using the modified HOG features proposed in [8]. Our training process proceeds in a way similar to the one described for the vehicle detector, except we do not consider rotations. We use exactly the same HOG parameters as the root model in [8], which we also compare against as a baseline approach. Windows have a size of $5 \times 15$ grid cells (same as the baseline), and each grid cell contains 32 features, for a total of 2400 features per window. As an initial training set, we randomly sample in scale and translation from the negative training images to obtain two negatives per image. We resize annotated training bounding boxes by their height, and also add a vertically flipped duplicate to the positive training set (we learn a single symmetric filter). We then train the classifiers, linear SVM, PLS, and CDF, setting parameters by 20-fold cross-validation. We consider up to 10 PLS factors for both CDF and PLS, and use QDA as the subsequent classifier. Once each classifier is trained, we perform sliding window detection, followed by non-maximal suppression, and hard-negative mining (up to 500 had negatives are added each iteration). Multi-scale detection, proceeds by sliding window on an image pyramid with 8 intervals per octave. As one of our baseline comparisons, we also include Felzenszwalb’s DPM Root model [8], which consists of two components (one for each facing direction), and was trained using Latent SVM, allowing the positives to differ from annotations to better align HOG features. While the comparison is unfair to our approach, which uses the positive bounding boxes exactly as provided, and groups all images into one model, Figure 7 shows that the improvement in discrimination due to our CDF model is enough to make up for the lack of a complex model. Also, it is worth noting that our approach outperforms the LDA model of Bharath et al. [11], on the same task, which achieves an AP of .75.

**Action Recognition: UCF50 + Action Bank.** We compare the performance of our method on multi-class action recognition using the UCF50 dataset [1]. We use the first 10 classes (out of a total of 50), to show our results. We use action bank features [24], which provide 14965 dimensional feature vectors for the UCF50 dataset. We perform 5-fold group-wise cross-validation as is done in [24]. We compare results using the following algorithms:

1) **Full features + linear-SVM:** linear SVM on the full 14965 dimensional feature vectors. This is the state-of-the-art reported by [24].
2) **Full features + RBF-SVM:** same as 1) but using an RBF SVM instead.
3) **PLS + RBF-SVM:** We obtained 1 to 16 PLS factors for all pairs of classes (10 classes, which led to 45 models), concatenated the factor scores for all pairs, and trained a RBF-SVM on the reduced feature space using a set of held out training samples.
4) **CDF + RBF-SVM:** same as 3) but instead of PLS we used 1 to 2 CDF composites.
5) **Linear-SVM + RBF-SVM:** Same as 3) but instead of PLS we used linear SVM. Each model has 1 SVM weight vector.

Table 2 shows the 5-fold group-wise cross-validation accuracy of each approach. CDF+RBF-SVM not only outperforms state of the art but it is also superior to all the other methods except for full features+RBF-SVM. CDF+RBF-SVM is significantly faster than full features+RBF-SVM during test time as shown in Table 2. Timing reported here might be unfair to full feature+linear SVM and full features+RBF-SVM. We have therefore also listed the number of high-dimensional(14965) operations\(^1\) (per test sample) required by each approach in Table 2.

\(^{1}\) # dot products for PLS+RBF-SVM, CDF+RBF-SVM linear-SVM+RBF-SVM, # 1-vs-rest classifiers for full-features+linear-SVM and #difference of vectors and norm for support vectors in case of full-features+RBF-SVM
Table 2: Accuracy and timings on the UCF50 Action Recognition datasets.

<table>
<thead>
<tr>
<th></th>
<th>PLS RBF-SVM</th>
<th>linear-SVM</th>
<th>CDF RBF-SVM</th>
<th>full-features linear-SVM</th>
<th>full-features RBF-SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>76.33</td>
<td>77.74</td>
<td>82.18</td>
<td>75.37</td>
<td>82.35</td>
</tr>
<tr>
<td>Timing (sec)</td>
<td>2.682</td>
<td>0.519</td>
<td>0.654</td>
<td>44.623</td>
<td>45.519</td>
</tr>
<tr>
<td>#High Dim Operations</td>
<td>306</td>
<td>45</td>
<td>64</td>
<td>10</td>
<td>799</td>
</tr>
</tbody>
</table>

Figure 5: Performance on Google 90° (top) and Google 45° (bottom) satellite imagery datasets. **Left:** Precision-recall curves comparing CDF to the baselines. **Center:** Back-projection of weight vector magnitudes computed by summing for each pixel the absolute values of the weights it contributed to. PLS captures many variations, but requires 8 or 9 factors. Linear SVM requires a single weight vector but captures mostly the contour of the car. CDF captures not only information about the contour of the car, but also the front and rear car windows. **Right:** True positives (TP), false positives (FP) and false negatives (FN) detected by the system. ©Google.

### 5 Discussion and Future work

We proposed and evaluated a new approach, CDF, which yields surprisingly good performance compared to PLS and SVM, and yields much more compact subspaces than PLS. The improvement is especially noticeable in the vehicle and human detection tasks, implying that CDF might be a good alternative to linear SVM for many state-of-the-art vision approaches. Our experiments, however, raise some questions that still need to be investigated. In particular, why do PLS and CDF seem to perform so well against linear SVM? This is still unclear, though we can see that the margin of improvement is much larger for the vision datasets than for the machine learning datasets. This might be due to the fact that samples away from the decision boundary have a positive significant contribution to the projection direction. This can be both an advantage and a disadvantage: more samples contributing to the projection direction can yield a better boundary, but only if the probability mass away from the boundary provides useful information. Other areas that deserve further investigation include the use of additional composite candidates (e.g., from SVM), other subsequent classifiers, and extension to a kernel method for applications where kernel methods are practical.
Figure 6: Sample vehicle detections for the Google 90° (left) and the Google 45° (right) datasets. Color represents the confidence of detection, red (high confidence) and blue (low confidence) being the two extremes. ©Google.

Figure 7: Left: Per image test-time sliding windows timings (in seconds) on Google 45°. Timings were taken on an Intel(R) Core(TM) i7-2620M CPU @ 2.70GHz with 6GB RAM. CDF is significantly faster than PLS and is 2 times slower than SVM, but as shown in Figure 5 CDF gives a precision of 60% as compared to 40% at 80% recall obtained by both PLS and SVM. Right: INRIA Pedestrian dataset performance. Precision-recall curves comparing CDF to the baselines. The comparison of cdf (our approach), to pls and svm (linear kernel) is fair, i.e., the classifiers are trained using exactly the same approach and input features. The comparison to dpm root is unfair to our approach, see text: Comparing dpm root to svm shows the impact of these additional improvements. Nevertheless, our single-component model without any of these improvements is able to outperform dpm root.

References


