Bilinear Classifiers for Visual Recognition

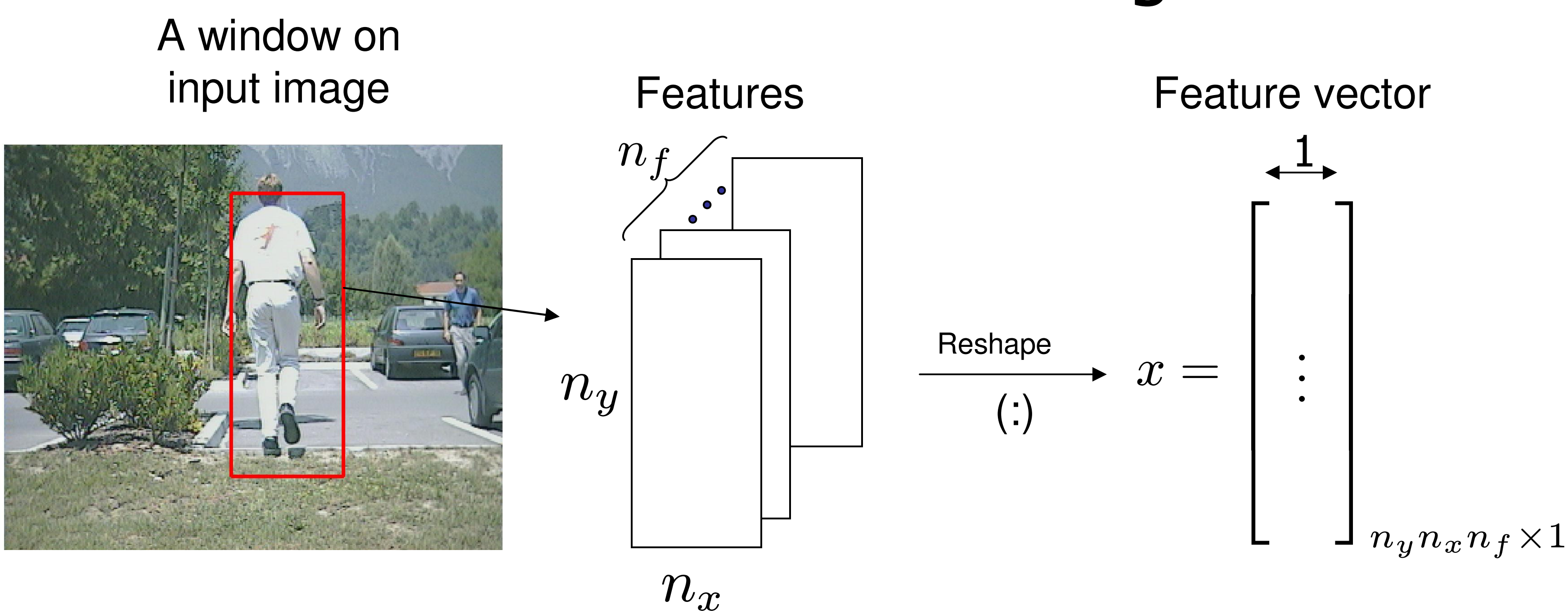
Hamed Pirsiavash   Deva Ramanan   Charless Fowlkes

Computational Vision Lab.
University of California Irvine

To be presented in NIPS 2009

Introduction

- Linear model for visual recognition



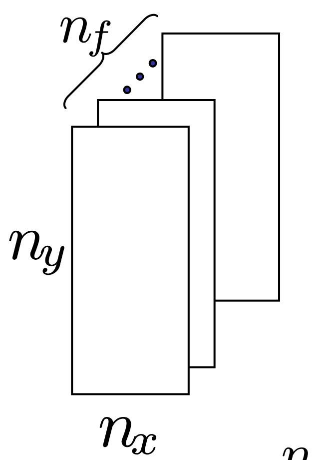
Introduction

- Linear classifier
  - Learn a template
  - Apply it to all possible windows

$$\underbrace{\begin{bmatrix} \\ \vdots \\ \\ \end{bmatrix}^T}_{\substack{\text{Template} \\ \text{(Model)}}} \underbrace{\begin{bmatrix} \\ \vdots \\ \\ \end{bmatrix}}_{\substack{\text{Feature} \\ \text{vector}}} > 0 \quad \Longleftrightarrow \quad w^T x > 0$$

Introduction

Features



$$\xrightarrow{\text{Reshape}} \quad X = \overset{n_f}{\begin{bmatrix} & \cdots \\ \vdots & \\ & \end{bmatrix}}_{n_y n_x \times n_f}$$

$$n_s := n_x n_y$$

$$\underbrace{\overset{n_f}{\begin{bmatrix} & \cdots \\ \vdots & \\ & \end{bmatrix}}_{n_s \times n_f}}_{W} = \underbrace{\overset{d}{\begin{bmatrix} \cdots \\ \vdots \\ \end{bmatrix}}_{n_s \times d}}_{W_s} \times \underbrace{\overset{n_f}{\begin{bmatrix} \vdots & \cdots \end{bmatrix}}_{d \times n_f}}_{W_f^T}$$

$$W = W_s W_f^T \qquad d \le \min(n_s, n_f)$$

Introduction

- Motivation for bilinear models
  - Reduced rank: less number of parameters
    - Better generalization: reduced over-fitting
    - Run-time efficiency
  - Transfer learning
    - Share a subset of parameters between different but related tasks

Outline

- Introduction
- Sliding window classifiers
- Bilinear model and its motivation
- Extension
- Related work
- Experiments
  - Pedestrian detection
  - Human action classification
- Conclusion

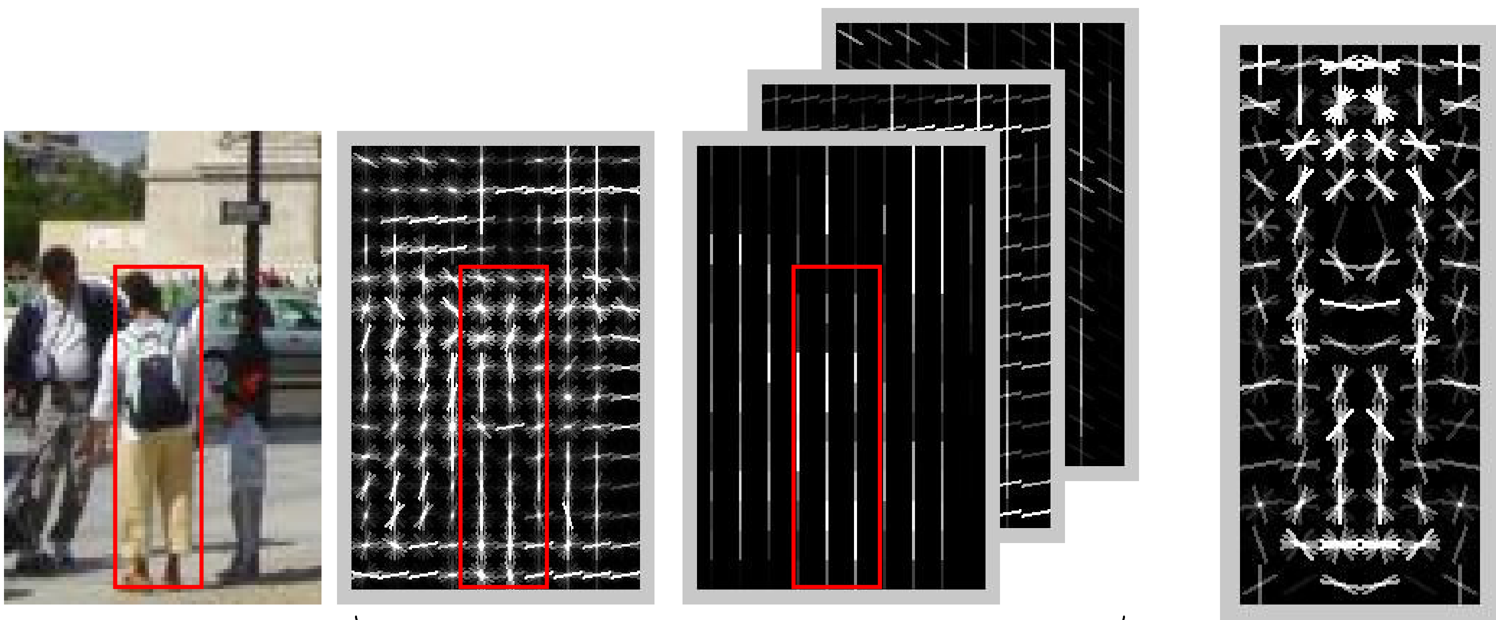
Sliding window classifiers

- Extract some visual features from a spatio-temporal window
  - e.g., histogram of gradients (HOG) in Dalal and Triggs' method
- Train a linear SVM using annotated positive and negative instances $w^T x > 0$

$$\min_w L(w) = \frac{1}{2} w^T w + C \sum_n \max(0, 1 - y_n w^T x_n)$$

- Detection: evaluate the model on all possible windows in space-scale domain
  - Use convolution since the model is linear

Sliding window classifiers



Sample image

Features

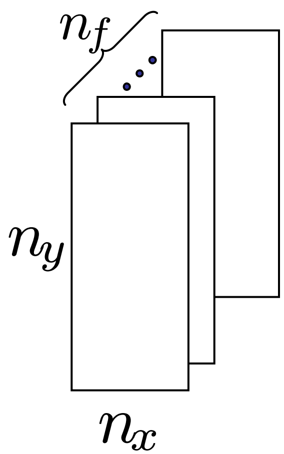
Sample template W

Bilinear model (Definition)

- Visual data are better modeled as matrices/tensors rather than vectors
  - Why not use the matrix structure

Bilinear model (Definition)

Features



$$\xrightarrow{\text{Reshape}}$$

$$X = \begin{array}{c} \overset{n_f}{\longleftrightarrow} \\ \begin{bmatrix} & \cdots & \\ \vdots & & \\ & & \end{bmatrix}_{n_s \times n_f} \end{array}$$

$$n_s := n_x n_y$$

$$Tr\Big(\underbrace{\overset{n_f}{\overleftrightarrow{\begin{bmatrix} & \cdots \\ \vdots & \\ & \end{bmatrix}}}^{T}}_{\text{Model } W} \underbrace{\overset{n_f}{\overleftrightarrow{\begin{bmatrix} & \cdots \\ \vdots & \\ & \end{bmatrix}}}}_{\text{Feature } X} \Big) = \underbrace{\begin{bmatrix} \\ \vdots \\ \\ \end{bmatrix}^{T}}_{w} \underbrace{\begin{bmatrix} \\ \vdots \\ \\ \end{bmatrix}}_{x}$$

$$\Rightarrow \quad Tr(W^T X) = w^T x$$

$$Tr(W^T X) > 0 \quad \Leftrightarrow \quad w^T x > 0$$

Bilinear model (Definition)

- Bilinear model

$$d \leq \min(n_s, n_f)$$

$$\underbrace{\overset{\xleftrightarrow{\quad n_f \quad}}{\begin{bmatrix} & \cdots & \\ \vdots & & \\ & & \end{bmatrix}}_{n_s \times n_f}}_{W} = \underbrace{\overset{\xleftrightarrow{\ d\ }}{\begin{bmatrix} \cdots \\ \vdots \\ \ \end{bmatrix}}_{n_s \times d}}_{W_s} \times \underbrace{\overset{\xleftrightarrow{\quad n_f \quad}}{\begin{bmatrix} \vdots & \cdots \end{bmatrix}}_{d \times n_f}}_{W_f^T}$$

$$W = W_s W_f^T$$

$w^T x > 0 \iff f(x) > 0$

$$f(X) = Tr(W_f W_s^T X)$$

Bilinear model (Definition)

- Bilinear model

$$d \leq \min(n_s, n_f)$$

$$\underbrace{\overset{\xleftrightarrow{\; n_f \;}}{\begin{bmatrix} & \cdots & \\ \vdots & & \\ & & \\ & & \end{bmatrix}}_{n_s \times n_f}}_{W} = \underbrace{\overset{\xleftrightarrow{\; d \;}}{\begin{bmatrix} \cdots \\ \vdots \\ \\ \end{bmatrix}}_{n_s \times d}}_{W_s} \times \underbrace{\overset{\xleftrightarrow{\; n_f \;}}{\begin{bmatrix} \vdots & \cdots \end{bmatrix}}_{d \times n_f}}_{W_f^T}$$

$$W = W_s W_f^T$$

$$w^T x > 0 \iff f(x) > 0 \qquad f(X) = Tr(W_f W_s^T X)$$

$$\Rightarrow f(X) = Tr(\underbrace{W_s^T X W_f}_{\text{Bilinear in } W_f \text{ and } W_s})$$

Bilinear model (Definition)

- Bilinear model $\qquad d \le \min(n_s, n_f)$

$$\underbrace{\overset{\xleftrightarrow{\;n_f\;}}{\begin{bmatrix} & \cdots & \\ \vdots & & \\ & & \end{bmatrix}}_{n_s \times n_f}}_{W} = \underbrace{\overset{\xleftrightarrow{\;d\;}}{\begin{bmatrix} \cdots \\ \vdots \\ \end{bmatrix}}_{n_s \times d}}_{W_s} \times \underbrace{\overset{\xleftrightarrow{\;n_f\;}}{\begin{bmatrix} \vdots & \cdots \end{bmatrix}}_{d \times n_f}}_{W_f^T}$$

$$W = W_s W_f^T$$

$w^T x > 0 \iff f(x) > 0$

$$f(X) = Tr(W_f W_s^T X)$$

$$\Rightarrow f(X) = Tr(\underbrace{W_s^T X W_f}_{\text{Bilinear in } W_f \text{ and } W_s})$$

Was Linear: $f(X) = Tr(W^T X)$

Bilinear model (Learning)

- Linear SVM for a given set of training pairs $\{X_n, y_n\}$

$$\underbrace{\min_W L(W)}_{\text{Objective function}} = \underbrace{\frac{1}{2}Tr(W^TW)}_{\text{Regularizer}} + \underbrace{C}_{\text{Parameter}} \underbrace{\sum_n \max(0, 1 - y_n Tr(W^T X_n))}_{\text{Constraints}}$$

Bilinear model (Learning)

- Linear SVM for a given set of training pairs $\{X_n, y_n\}$

$$\min_W \underbrace{L(W)}_{\text{Objective function}} = \underbrace{\tfrac{1}{2}Tr(W^T W)}_{\text{Regularizer}} + C \underbrace{\sum_n \max(0, 1 - y_n Tr(W^T X_n))}_{\text{Constraints}}$$

$W := W_s W_f^T$

Parameter

Bilinear model (Learning)

- For Bilinear formulation

$$\min L(W_s, W_f) = \frac{1}{2} Tr(W_f W_s^T W_s W_f^T) + C \sum_n \max(0, 1 - y_n Tr(W_f W_s^T X_n))$$

- Biconvex so solve by coordinate decent
  - By fixing one set of parameters, it's a typical SVM problem (with a change of basis)
  - Use off-the-shelf SVM solver in the loop

Motivation

- Regularization
  - Similar to PCA, but not orthogonal and learned discriminatively and jointly with the template

$$W = \underbrace{W_s}_{\text{Reduced dimensional Template}} \underbrace{W_f^T}_{\text{Subspace}}$$

- Run-time efficiency
  - d convolutions instead of n_f

$$d \leq \min(n_s, n_f)$$

Motivation

- Transfer learning $W = W_s W_f^T$
  - Share the subspace W_f between different problems
    - e.g human detector and cat detector
  - Optimize the summation of all objective functions
    - Learn a good subspace using all data

Extension

- Multi-linear
  - High-order tensors
    - $L(W_x, W_y, W_f)$ instead of just $L(W_s, W_f)$
    - For 1D feature $L(W_x, W_y)$
      - Separable filter for (Rank=1)
    - Spatio-temporal templates

$L(W_x, W_y, W_t, W_f)$

Related work (Rank restriction)

- Bilinear models
  - Often used in increasing the flexibility; however, we use them to reduce the parameters.
  - Mostly used in generative models like density estimation and we use in classification
- Soft Rank restriction
  - They used $Tr(W)$ rather than $Tr(W^T W)$ in SVM to regularize on rank
    - Convex, but not easy to solve
    - Decrease summation of eigen values instead of the number of non-zero eigen values (rank)
- Wolf et al (CVPR'07)
  - Used a formulation similar to ours with hard rank restriction
  - Showed results only for soft rank restriction
  - Used it only for one task (Didn't consider multi-task learning)

Related work (Transfer learning)

- Dates back to at least Caruana's work (1997)
  - We got inspired by their work on multi-task learning
  - Worked on: Back-propagation nets and k-nearest neighbor
- Ando and Zhang's work (2005)
  - Linear model
  - All models share a component in low-dimensional subspace (transfer)
  - Use the same number of parameters

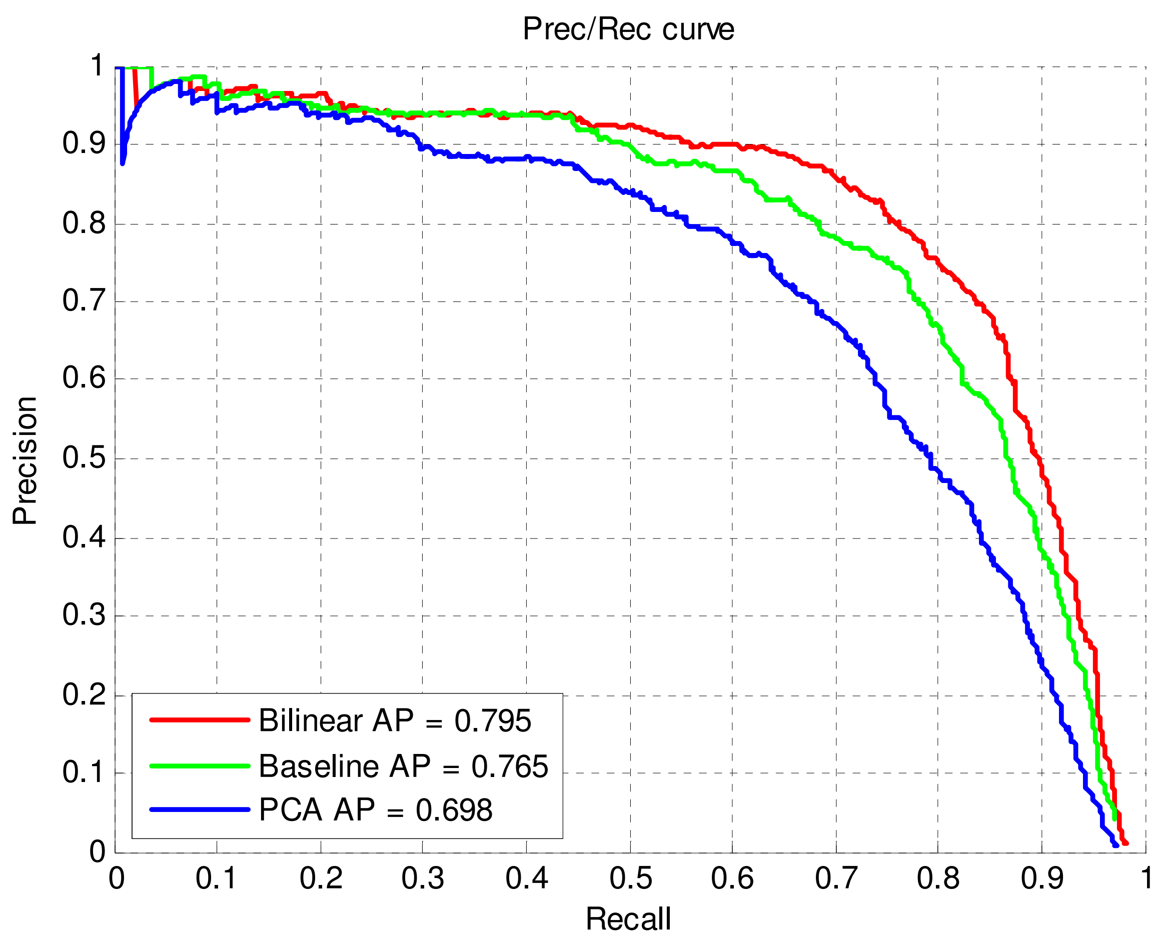
Experiments: Pedestrian detection

- Baseline: Dalal and Triggs' spatio-temporal classifier (ECCV'06)
  - Linear SVM on features: (84 for each 8×8 cell)
    - Histogram of gradient (HOG)
    - Histogram of optical flow
  - Made sure that the spatiotemporal is better than the static one by modifying the learning method

Experiments: Pedestrian detection

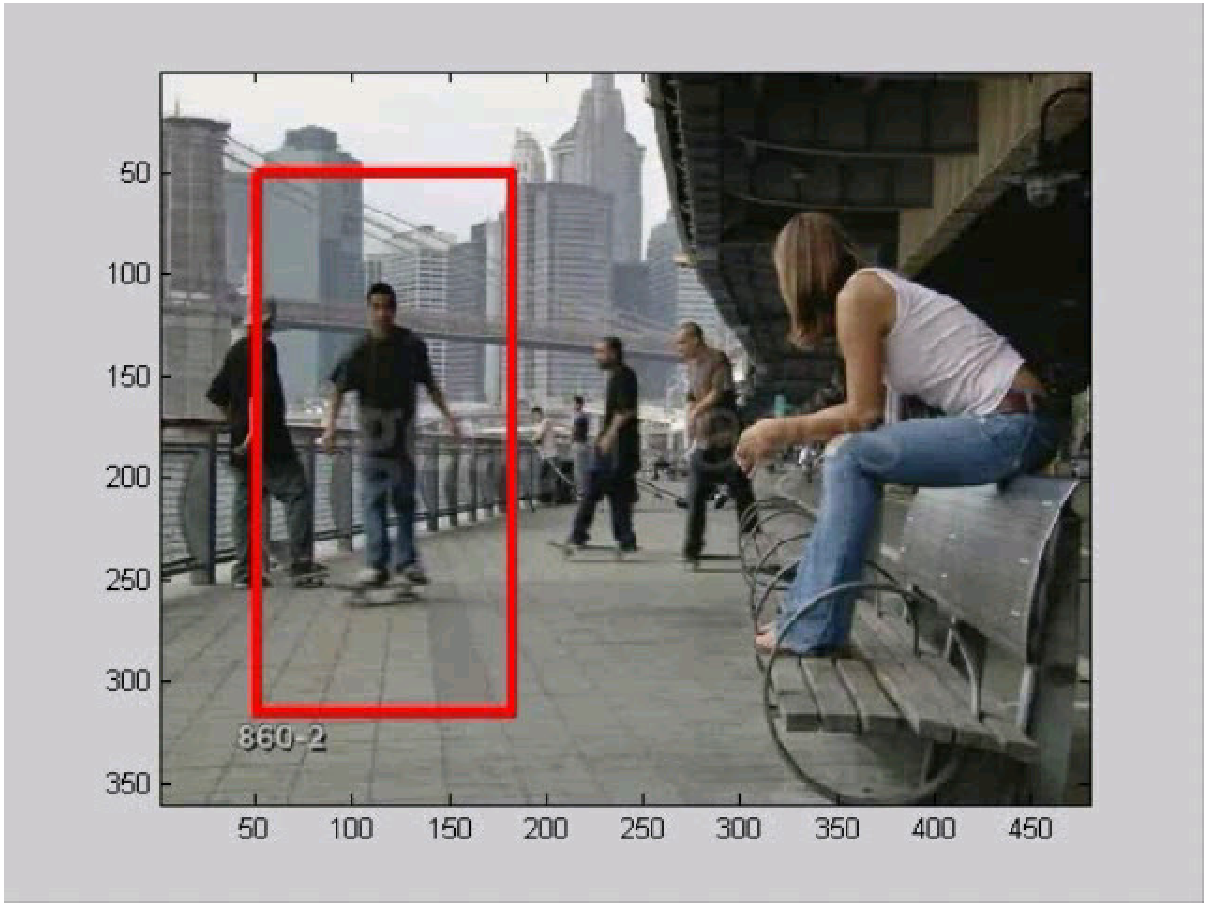
- Dataset: INRIA motion and INRIA static
  - 3400 video frame pairs
  - 3500 static images
- Typical values:
  - $n_s = 14 \times 6,\ n_f = 84,\ d = 5 \text{ or } 10$
- Evaluation
  - Average precision
- Initialize with PCA in feature space
- Ours is 10 times faster

Experiments: Pedestrian detection



Experiments: Pedestrian detection

Experiments: Pedestrian detection

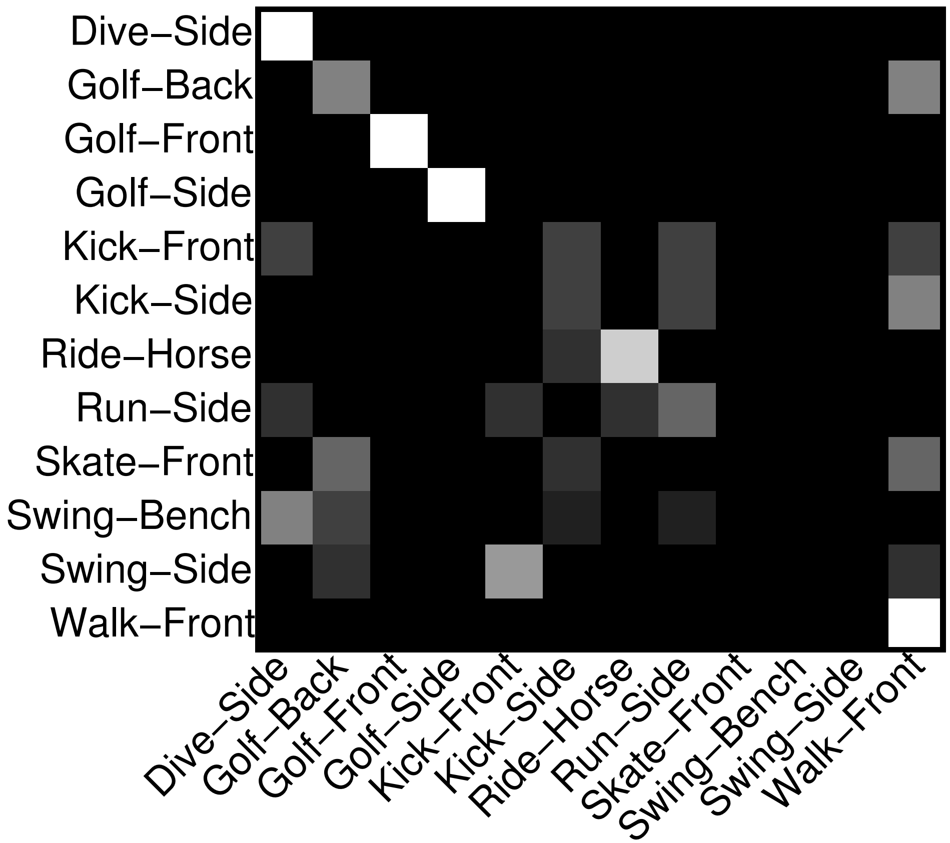


Link

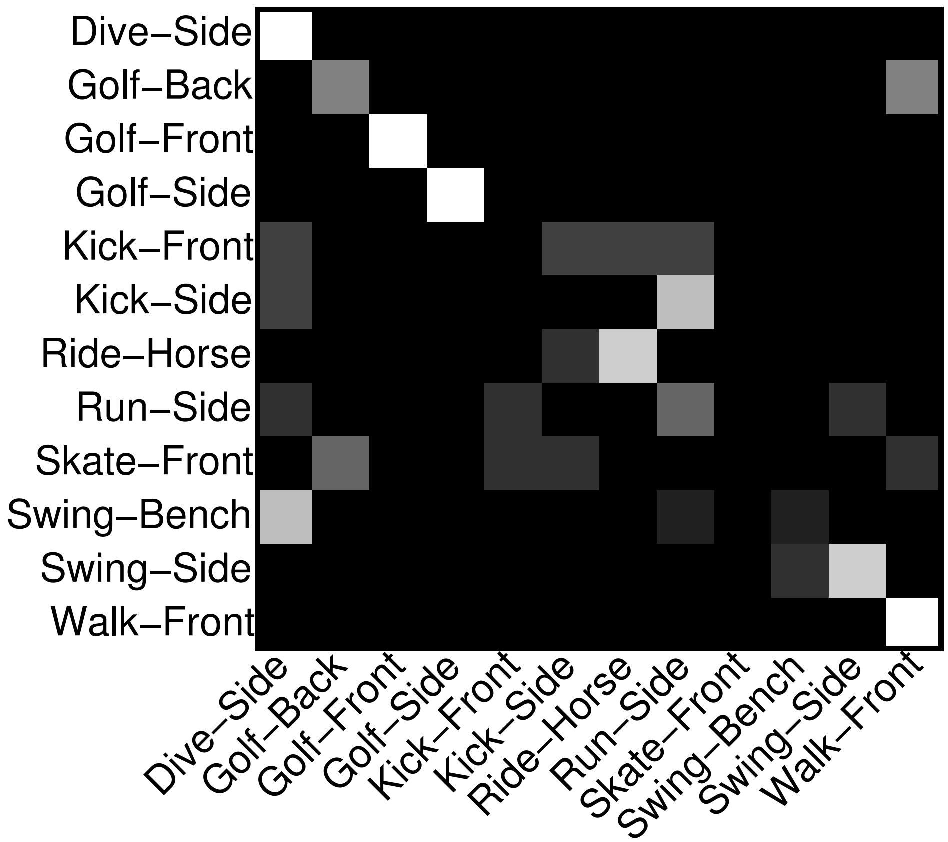
Experiments: Human action classification 1

- 1 vs all action templates
- Voting:
  - A second SVM on confidence values
- Dataset:
  - UCF Sports Action (CVPR 2008)
  - They obtained 69.2%
  - We got 64.8% but
    - More classes: 12 classes rather than 9
    - Smaller dataset: 150 videos rather than 200
    - Harder evaluation protocol: 2-fold vs. LOOCV
    - 87 training examples rather than 199 in their case

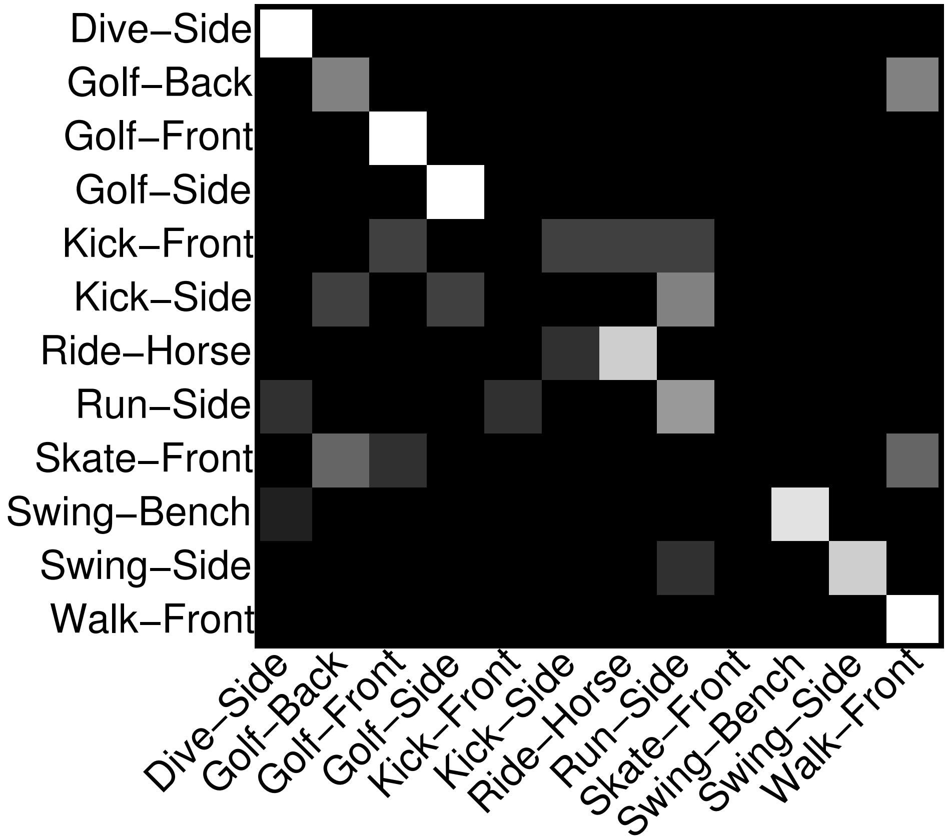
UCF action Results: PCA (0.444)



UCF action Results: Linear (0.518)

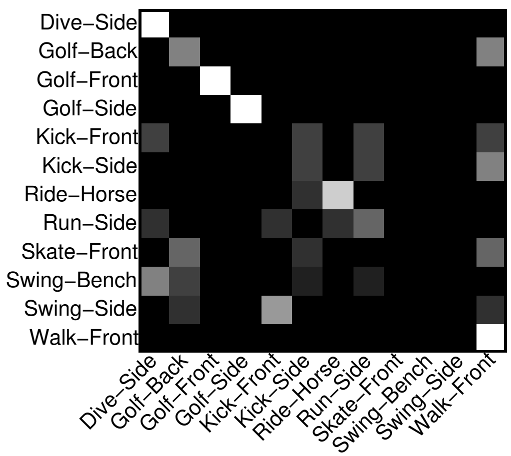


UCF action Results: Bilinear (0.648)



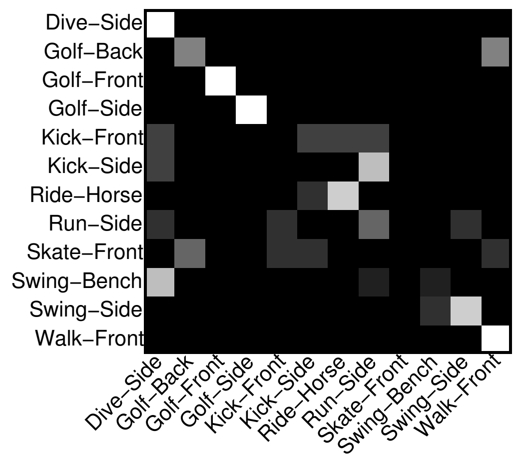
UCF action Results

PCA on features (0.444)

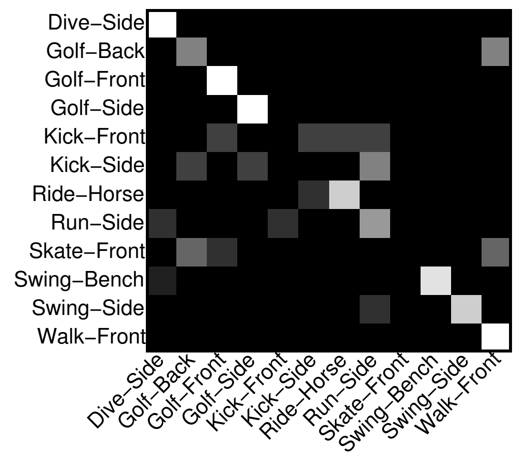


Linear (0.518)

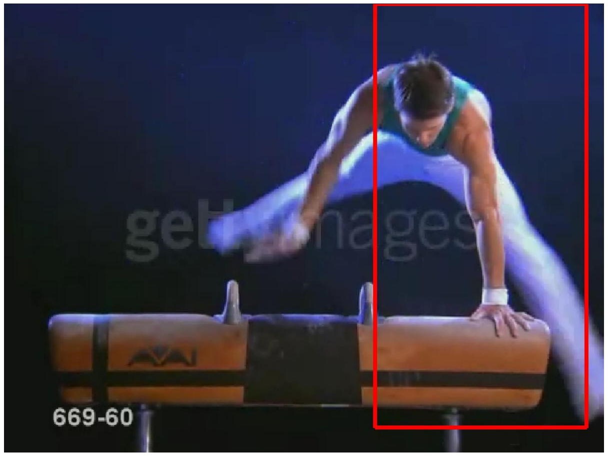
(Not always feasible)



Bilinear (0.648)



UCF action Results

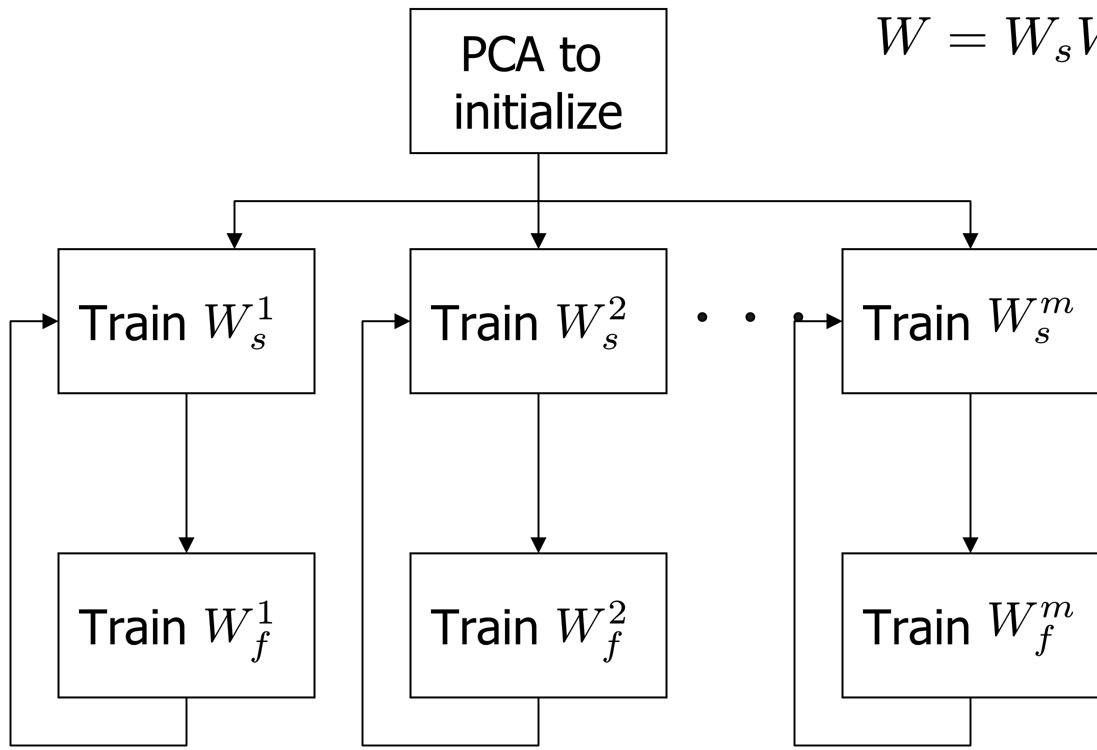


Experiments: Human action classification 2

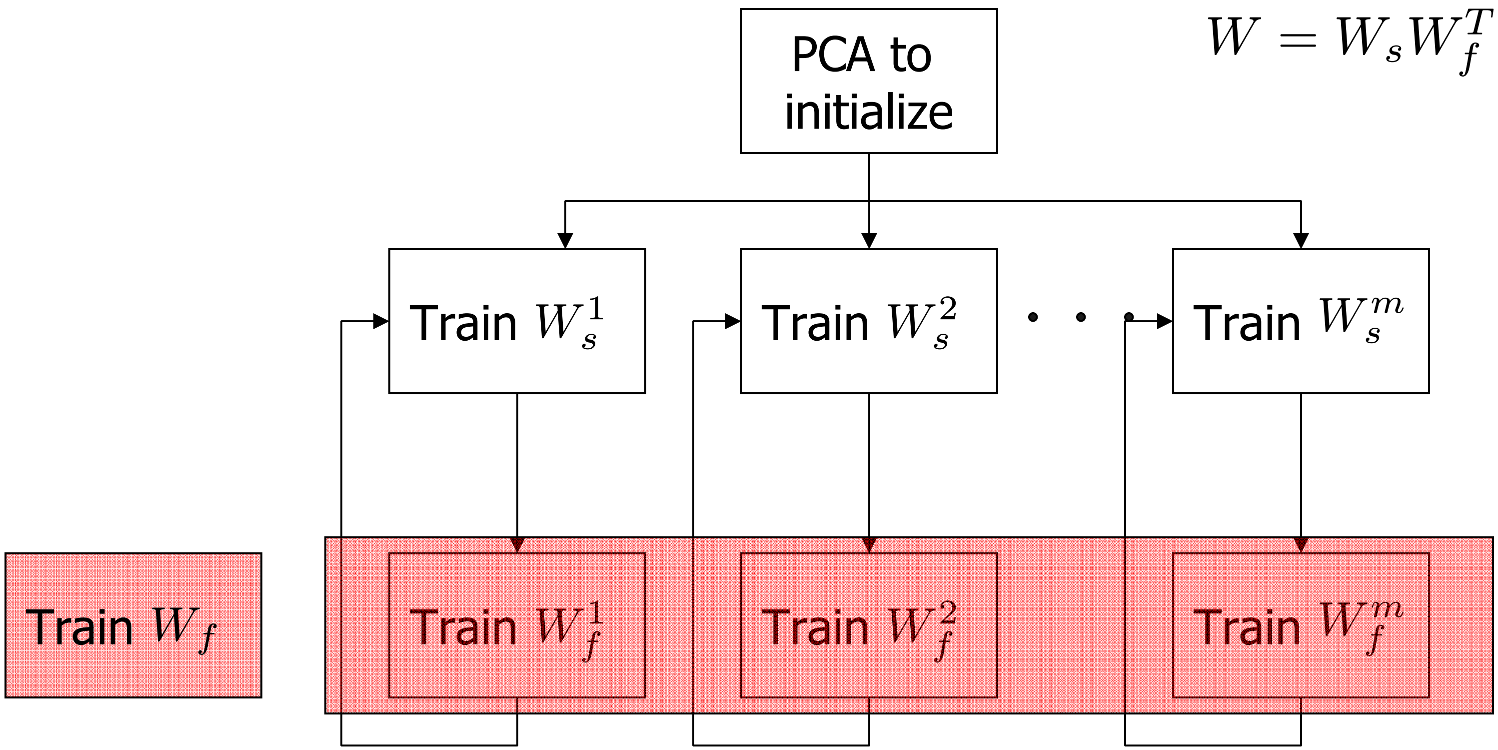
- Transfer
  - We used only two examples for each of 12 action classes
  - Once trained independently
  - Then trained jointly
    - Shared the subspace
    - Adjusted the C parameter for best result

Transfer

$$W = W_s W_f^T$$



Transfer



$$\min_{W_f} \sum_{i=1}^{m} L(W_f, W_s^i)$$

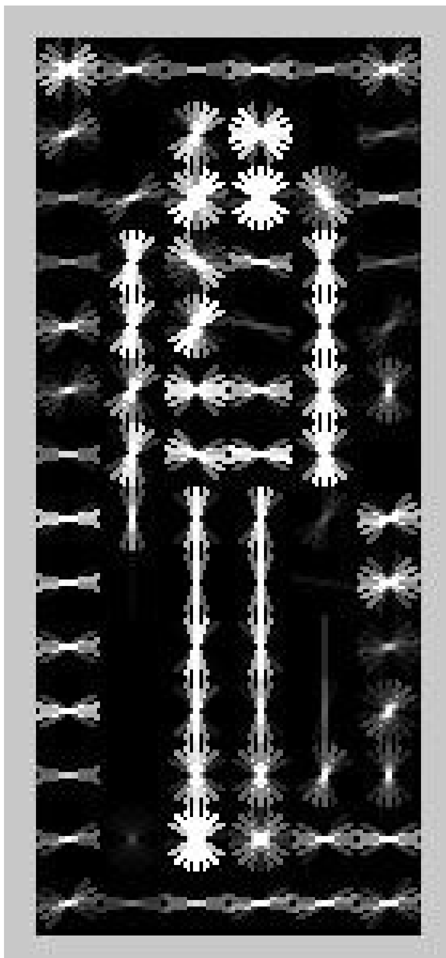
Results: Transfer

Average classification rate

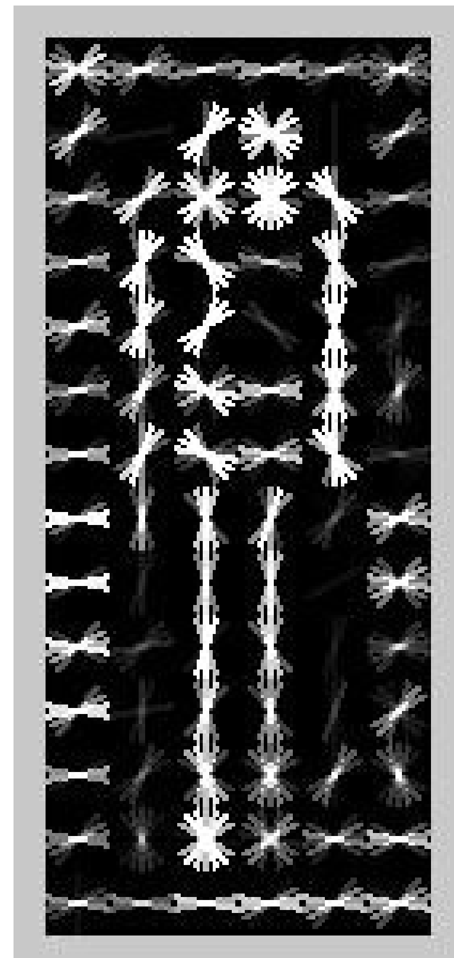
	Coordinate decent iteration 1	Coordinate decent iteration 2
Independent bilinear (C=.01)	0.222	0.289
Joint bilinear (C=.1)	0.269	0.356

Results: Transfer (for "walking")

Iteration 1



Refined at Iteration 2



Conclusion

- Introduced multi-linear classifiers
  - Exploit natural matrix/tensor representation of spatio-temporal data
- Trained with existing efficient linear solvers
- Shared subspace for different problems
  - A novel form of transfer learning
- Got better performance and about 10X speed up in run-time compared to the linear classifier.
- Easy to apply to most high dimensional features (instead of dimensionality reduction methods like PCA)
- Simple: ~ 20 lines of Matlab code

Thanks!

Bilinear model (Learning details)

- Linear SVM for a given set of training pairs $\{x_n, y_n\}$

$$\min_W L(W) = \frac{1}{2}Tr(W^T W) + C\sum_n \max(0, 1 - y_n Tr(W^T X_n))$$

- For Bilinear formulation

$$\min L(W_f, W_s) = \frac{1}{2}Tr(W_f W_s^T W_s W_f^T) + C\sum_n \max(0, 1 - y_n Tr(W_f W_s^T X_n))$$

- It is biconvex so solve by coordinate decent

Bilinear model (Learning details)

- Each coordinate descent iteration:

  freeze W_s

$$\min_{\tilde{W}_f} L(\tilde{W}_f, W_s) = \frac{1}{2}(\tilde{W}_f^T \tilde{W}_f) + C \sum_n \max(0, 1 - y_n Tr(\tilde{W}_f^T \tilde{X}_n))$$

  where $\tilde{W}_f = A_s^{\frac{1}{2}} W_f^T, \; \tilde{X}_n = A_s^{-\frac{1}{2}} W_s^T X_n, \; A_s = W_s^T W_s$

  then freeze W_f

$$\min_{\tilde{W}_s} L(W_f, \tilde{W}_s) = \frac{1}{2}(\tilde{W}_s^T \tilde{W}_s) + C \sum_n \max(0, 1 - y_n Tr(\tilde{W}_s^T \tilde{X}_n))$$

  where $\tilde{W}_s = W_s A_f^{\frac{1}{2}}, \; \tilde{X}_n = X_n W_f A_f^{-\frac{1}{2}}, \; A_f = W_f^T W_f$