C280, Computer Vision

Prof. Trevor Darrell
trevor@eecs.berkeley.edu

Lecture 14: Discriminative Kernels for Recognition
Two Lectures ago...

- Scanning window paradigm
- GIST
- HOG
- Boosted Face Detection
- Local-feature Alignment; from Roberts to Lowe...
- BOW Indexing
Last Lecture: Topic Models for Recognition

Guest lecture by Kate Saenko:

- Dataset issues
- Topic models for category discovery [Sivic05]
- Category discovery from web [Fergus05]
- Bootstrapping a category model [Li07]
- Using text in addition to image [Berg06]
- Learning objects from a dictionary [Saenko08]
Last Lecture Summary

• The web contains unlimited, but extremely noisy object category data

• The text surrounding the image on the web page is an important recognition cue

• Topic models (pLSA, LDA, HDP, etc.) are useful for discovering objects in images and object senses in text

• Bootstrap model from small amount of labeled or weakly labeled data

• Still an open research problem!
Today: Discriminative Kernels

- SVM-BOW
- Pyramid and Spatial-Pyramid match
- Fast Intersection Kernels
- Latent-part SVM models
Object categorization: the statistical viewpoint

\[ p(\text{zebra} \mid \text{image}) \]

vs.

\[ p(\text{no zebra} \mid \text{image}) \]

- Bayes rule:

\[
\frac{p(\text{zebra} \mid \text{image})}{p(\text{no zebra} \mid \text{image})} = \frac{p(\text{image} \mid \text{zebra})}{p(\text{image} \mid \text{no zebra})} \cdot \frac{p(\text{zebra})}{p(\text{no zebra})}
\]

- Posterior ratio
- Likelihood ratio
- Prior ratio

Fei-Fei, Fergus, Torralba, CVPR 2007
Object categorization: the statistical viewpoint

\[
\frac{p(\text{zebra} \mid \text{image})}{p(\text{no zebra} \mid \text{image})} = \frac{p(\text{image} \mid \text{zebra})}{p(\text{image} \mid \text{no zebra})} \cdot \frac{p(\text{zebra})}{p(\text{no zebra})}
\]

- **Posterior ratio**
- **Likelihood ratio**
- **Prior ratio**

- **Discriminative methods** model posterior
- **Generative methods** model likelihood and prior

Fei-Fei, Fergus, Torralba, CVPR 2007 SC
Discriminative

- Direct modeling of \( \frac{p(zebra \mid image)}{p(no\ zebra \mid image)} \)
Generative

- Model $p(image | zebra)$ and $p(image | no zebra)$
1. Feature detection and representation

• Regular grid
  – Vogel & Schiele, 2003
  – Fei-Fei & Perona, 2005

• Interest point detector
  – Csurka, Bray, Dance & Fan, 2004
  – Fei-Fei & Perona, 2005
  – Sivic, Russell, Efros, Freeman & Zisserman, 2005

• Other methods
  – Random sampling (Vidal-Naquet & Ullman, 2002)
  – Segmentation based patches (Barnard, Duygulu, Forsyth, de Freitas, Blei, Jordan, 2003)
1. Feature detection and representation

Detect patches

- [Mikojaczyk and Schmid '02]
- [Mata, Chum, Urban & Pajdla, '02]
- [Sivic & Zisserman, '03]

Compute SIFT descriptor

[Lowe'99]

Normalize patch

Slide credit: Josef Sivic
1. Feature detection and representation

Fei-Fei, Fergus, Torralba, CVPR 2007 SC
2. Codewords dictionary formation

Fei-Fei, Fergus, Torralba, CVPR 2007 SC
2. Codewords dictionary formation

Vector quantization

Slide credit: Josef Sivic
2. Codewords dictionary formation

Fei-Fei et al. 2005
Image patch examples of codewords

Sivic et al. 2005
3. Image representation

![Diagram showing frequency of codewords](image)

Fei-Fei, Fergus, Torralba, CVPR 2007 SC
1. feature detection & representation

2. codewords dictionary

3.
Learning and Recognition

codewords dictionary

category models (and/or) classifiers

category decision
Learning and Recognition

1. Generative method:
   - graphical models

2. Discriminative method:
   - SVM

Fei-Fei, Fergus, Torralba, CVPR 2007 SC
Discriminative methods based on ‘bag of words’ representation
Dance et al.

Detect or sample features

List of positions, scales, orientations

Describe features

Associated list of d-dimensional descriptors

Quantize to form bag of words vector for the image

SVM
Visual Categorization with Bags of Keypoints

Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, Cédric Bray

Xerox Research Centre Europe
6, chemin de Maupertuis
38240 Meylan, France
{gcsurka,cdance}@xrc-e.xerox.com

Abstract. We present a novel method for generic visual categorization: the problem of identifying the object content of natural images while generalizing across variations inherent to the object class. This bag of keypoints method is based on vector quantization of affine invariant descriptors of image patches. We propose and compare two alternative implementations using different classifiers: Naïve Bayes and SVM. The main advantages of the method are that it is simple, computationally efficient and intrinsically invariant. We present results for simultaneously classifying seven semantic visual categories. These results clearly demonstrate that the method is robust to background clutter and produces good categorization accuracy even without exploiting geometric information.

Embedding

**Figure 1.6** The idea of SVMs: map the training data into a higher-dimensional feature space via $\Phi$, and construct a separating hyperplane with maximum margin there. This yields a nonlinear decision boundary in input space. By the use of a kernel function (1.2), it is possible to compute the separating hyperplane without explicitly carrying out the map into the feature space.
Kernels

• linear classifier:

\[ f(x) = \text{sign}(w^T x + b) \]

• Kernel classifier:

\[ K(u, v) = \Phi(u) \cdot \Phi(v) \]

\[ f(x) = \text{sign} \left( \sum_i y_i \alpha_i K(x, x_i) + b \right). \]

[Dance et al.]
Figure 1.7 Example of an SV classifier found using a radial basis function kernel $k(x, x') = \exp(-\|x - x'\|^2)$ (here, the input space is $\mathcal{X} = [-1, 1]^2$). Circles and disks are two classes of training examples; the middle line is the decision surface; the outer lines precisely meet the constraint (1.25). Note that the SVs found by the algorithm (marked by extra circles) are not centers of clusters, but examples which are critical for the given classification task. Gray values code $|\sum_{i=1}^m y_i \alpha_i k(x, x_i) + b|$, the modulus of the argument of the decision function (1.35). The top and the bottom lines indicate places where it takes the value 1 (from [471]).
Tried linear, quadratic, cubic; linear had best performance....

\[ K(\text{image, text}) = K(\text{image, text}) = \langle \text{features}, \text{features} \rangle \]

[Dance et al.]
Fig. 5. Images correctly classified containing multiple objects of the same category.

Table 2. Confusion matrix and mean rank for SVM (k=1000, linear kernel).

<table>
<thead>
<tr>
<th>True classes</th>
<th>faces</th>
<th>buildings</th>
<th>trees</th>
<th>cars</th>
<th>phones</th>
<th>bikes</th>
<th>books</th>
</tr>
</thead>
<tbody>
<tr>
<td>faces</td>
<td>98</td>
<td>14</td>
<td>10</td>
<td>10</td>
<td>34</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>buildings</td>
<td>1</td>
<td>63</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>trees</td>
<td>1</td>
<td>10</td>
<td>81</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>cars</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>85</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>phones</td>
<td>0</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>55</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>bikes</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>91</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>books</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>73</td>
</tr>
<tr>
<td>Mean ranks</td>
<td>1.04</td>
<td>1.77</td>
<td>1.28</td>
<td>1.30</td>
<td>1.83</td>
<td>1.09</td>
<td>1.39</td>
</tr>
</tbody>
</table>

Table 1. Confusion matrix and the mean rank for the best vocabulary (k=1000).

<table>
<thead>
<tr>
<th>True classes</th>
<th>faces</th>
<th>buildings</th>
<th>trees</th>
<th>cars</th>
<th>phones</th>
<th>bikes</th>
<th>books</th>
</tr>
</thead>
<tbody>
<tr>
<td>faces</td>
<td>76</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>buildings</td>
<td>2</td>
<td>44</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>trees</td>
<td>3</td>
<td>2</td>
<td>80</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>cars</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>75</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>phones</td>
<td>9</td>
<td>15</td>
<td>1</td>
<td>16</td>
<td>70</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>bikes</td>
<td>2</td>
<td>15</td>
<td>12</td>
<td>0</td>
<td>8</td>
<td>73</td>
<td>0</td>
</tr>
<tr>
<td>books</td>
<td>4</td>
<td>19</td>
<td>0</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>69</td>
</tr>
<tr>
<td>Mean ranks</td>
<td>1.49</td>
<td>1.88</td>
<td>1.33</td>
<td>1.33</td>
<td>1.63</td>
<td>1.57</td>
<td>1.57</td>
</tr>
</tbody>
</table>
How to Compare Sets of Features?

• Each instance is unordered set of vectors
• Varying number of vectors per instance

\[ X = \{ \vec{x}_1, \ldots, \vec{x}_m \} \]

\[ Y = \{ \vec{y}_1, \ldots, \vec{y}_n \} \]
Recap: Conventional Approaches

“Voting” – for each patch, find the most similar patch in database, and vote for the image containing that patch.

[Schmid, Lowe, Tuytelaars et al.]
Recap: Conventional Approaches

“Voting” – for each patch, find the most similar patch in database, and vote for the image containing that patch.  

[Schmid, Lowe, Tuytelaars et al.]

“Bag of Words” – quantize descriptor space, represent each image as a histogram over prototypes; use L1 indexing or SVM recognition.  

[Csurka et al., Sivic & Zisserman, Lazebnik & Ponce, Agarwal & Triggs]
Recap: Conventional Approaches

“Voting” – for each patch in database, and vote for patch.

Ignores co-occurrence; can be costly; works well for instance matching.

[Schmid, Lowe, Tuytelaars et al.]

“Bag of Words” – quantize descriptor space, represent each image as a histogram over prototypes; use L1 indexing or SVM recognition.

Sensitive to choice of quantization: how many “visual words”? - - -

[Csurka et al., Sivic & Zisserman, Lazebnik & Ponce, Agarwal & Triggs]
How Many Visual Words?

[Grauman and Darrell, CVPR 2005]
Correspondence-Based Match

Explicit search for correspondences...

$$\max_{\pi: X \rightarrow Y} \sum_{x_i \in X} S(x_i, \pi(x_i))$$

[Wallraven et al., Lyu, Boughhorbel et al., Belongie et al., Rubner et al., Berg et al., Gold & Rangarajan, Shashua & Hazan, …]
Partial Matching

Compare sets by computing a *partial matching* between their features.

\[
\max_{\pi: X \to Y} \sum_{x_i \in X} S(x_i, \pi(x_i))
\]
Pyramid Match

\[ X = \{ \tilde{x}_1, \ldots, \tilde{x}_m \} \quad \text{and} \quad Y = \{ \tilde{y}_1, \ldots, \tilde{y}_n \} \]

\[ \max_{\pi: X \to Y} \sum_{x_i \in X} S(x_i, \pi(x_i)) \]

MIT CSAIL Vision interfaces
Computing the Partial Matching

- Optimal matching $O(dm^3)$
- Greedy matching $O(dm^2 \log m)$
- Pyramid match $O(dmL)$

for sets with $O(m)$ features of dimension $d$

[Grauman and Darrell, ICCV 2005, JMLR 2007]
Pyramid Match Overview

Pyramid match measures similarity of a partial matching between two sets:

- Place multi-dimensional, multi-resolution grid over point sets
- Consider points matched at finest resolution where they fall into same grid cell
- Approximate optimal similarity with worst case similarity within pyramid cell

No explicit search for matches!
Pyramid match: main idea

Feature space partitions serve to “match” the local descriptors within successively wider regions.

$X = \{ \bar{x}_1, \ldots, \bar{x}_m \} \quad Y = \{ \bar{y}_1, \ldots, \bar{y}_n \}$
Pyramid match: main idea

$X = \{x_1, \ldots, x_m\}$  \hspace{1cm}  $Y = \{y_1, \ldots, y_n\}$

$\mathbb{R}^d$

Histogram intersection counts number of possible matches at a given partitioning.

$I(H_X, H_Y) = \sum_j \min(H_X(j), H_Y(j))$

= 3
Pyramid match kernel

\[
K_{\Delta}(X, Y) = \sum_{i=0}^{L} 2^{-i} \mathcal{I} \left( H_X^{(i)}, H_Y^{(i)} \right) - \mathcal{I} \left( H_X^{(i-1)}, H_Y^{(i-1)} \right)
\]

- For similarity, weights inversely proportional to bin size (or may be learned)
- Normalize these kernel values to avoid favoring large sets

[Grauman & Darrell, ICCV 2005]
Pyramid match kernel

$$X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \quad Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$$

Optimal match: $O(m^3)$
Pyramid match: $O(mL)$

optimal partial matching
Highlights of the pyramid match

- Linear time complexity
- Formal bounds on expected error
- Mercer kernel
- Data-driven partitions allow accurate matches even in high-dim. feature spaces
- Strong performance on benchmark object recognition datasets
Recognition results: Caltech-101 dataset

- 101 categories
- 40-800 images per class

Data provided by Fei-Fei, Fergus, and Perona
Pyramid match recognition on the Caltech-101

Pyramid match kernel (PMK)

Bag-of-words baseline

Recognition accuracy per class (10 runs)

Number of training examples per class
Recognition results:
Caltech-101 dataset

- Jain, Huynh, & Grauman (2007)
- Grauman & Darrell (2005)
- Wang et al. (2006)
- Serre et al. (2007)
- Frome et al. (2007)
- Zhang et al. (2006)
- Berg (2005)
- Berg et al. (2005)
- Holub et al. (2005)
Recognition results: Caltech-101 dataset

Combination of pyramid match and correspondence kernels

Accuracy

Number of training examples

[Kapoor et al. IJCV 2009]
Pyramid match kernel: examples of extensions and applications by other groups

- **Spatial Pyramid Match Kernel**
  - Scene recognition

- **Representing Shape with a Pyramid Kernel**
  - Shape representation

- **Dual-space Pyramid Matching**
  - Hu et al., 2007.
  - Medical image classification
Pyramid match kernel: examples of extensions and applications by other groups


From Omnidirectional Images to Hierarchical Localization, Murillo et al. 2007.

Action recognition  Video indexing  Robot localization
**Vocabulary-guided pyramid match**

- Tune pyramid partitions to the feature distribution
- Accurate for $d > 100$
- Requires initial corpus of features to determine pyramid structure
- Small cost increase over uniform bins: $kL$ distances against bin centers to insert points

---

[Uniform bins]

[Uniform bins]

[Vocabulary-guided bins]

[Vocabulary-guided bins]

---

[Grauman & Darrell, NIPS 2006]
Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories

Svetlana Lazebnik, Cordelia Schmid, Jean Ponce

Slides by: Lubomir Bourdev
Slide credits: Svetlana Lazebnik
Key Idea

• Pyramid Match Kernel (Grauman & Darrell)
  Pyramid in feature space, ignore location

• Spatial Pyramid (this work)
  Pyramid in image space, quantize features
Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Create intersection kernels
5. Train an SVM
Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Create intersection kernels
5. Train an SVM

OR

Weak (edge orientations)  OR  Strong (SIFT)
Algorithm

1. Extract interest point descriptors (dense scan)
2. **Construct visual word dictionary**
3. Build spatial histograms
4. Create intersection kernels
5. Train an SVM

- Vector quantization
- Usually K-means clustering
- Vocabulary size (16 to 400)
Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Create intersection kernels
5. Train an SVM
Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Create intersection kernels
5. Train an SVM

Feature histograms:
Level 3

Level 2

Level 1

Level 0

Total weight (value of pyramid match kernel): $I_3 + \frac{1}{2}(I_2 - I_3) + \frac{1}{4}(I_1 - I_2) + \frac{1}{8}(I_0 - I_1)$
Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Create intersection kernels
5. Train an SVM
Scene category dataset

Fei-Fei & Perona (2005), Oliva & Torralba (2001)

http://www-cvr.ai.uiuc.edu/ponce_grp/data

Multi-class classification results (100 training images per class)

<table>
<thead>
<tr>
<th>Level</th>
<th>Weak features (vocabulary size: 16)</th>
<th>Strong features (vocabulary size: 200)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single-level</td>
<td>Pyramid</td>
</tr>
<tr>
<td>0 (1 × 1)</td>
<td>45.3 ±0.5</td>
<td></td>
</tr>
<tr>
<td>1 (2 × 2)</td>
<td>53.6 ±0.3</td>
<td>56.2 ±0.6</td>
</tr>
<tr>
<td>2 (4 × 4)</td>
<td>61.7 ±0.6</td>
<td>64.7 ±0.7</td>
</tr>
<tr>
<td>3 (8 × 8)</td>
<td>63.3 ±0.8</td>
<td><strong>66.8 ±0.6</strong></td>
</tr>
</tbody>
</table>

Fei Fei & Perona: 65.2%
Scene category retrieval

Query
- Kitchen
- Kitchen
- Store
- Tall bldg
- Tall bldg
- Inside city

Retrieved images
- Living room
- Living room
- Living room
- Office
- Living room
- Living room
- Living room
- Inside city
- Mountain
- Forest
- Inside city
- Inside city
- Inside city
- Tall bldg
Scene category confusions

Difficult indoor images

kitchen  living room  bedroom
Caltech101 dataset

Fei-Fei et al. (2004)


Multi-class classification results (30 training images per class)

<table>
<thead>
<tr>
<th>Level</th>
<th>Weak features (16)</th>
<th>Strong features (200)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single-level</td>
<td>Pyramid</td>
</tr>
<tr>
<td>0</td>
<td>15.5 ±0.9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>31.4 ±1.2</td>
<td>32.8 ±1.3</td>
</tr>
<tr>
<td>2</td>
<td>47.2 ±1.1</td>
<td>49.3 ±1.4</td>
</tr>
<tr>
<td>3</td>
<td>52.2 ±0.8</td>
<td>54.0 ±1.1</td>
</tr>
</tbody>
</table>
Caltech101 comparison

Zhang, Berg, Maire & Malik, 2006

The diagram shows the mean recognition rate per class as a function of the number of training examples per class. The graph compares different methods, with our method indicated by an arrow. The legend lists various methods and their sources, such as Zhang, Berg, Maire & Malik (CVPR06), Lazebnik, Schmid & Ponce (CVPR06), and others.
Caltech101 challenges

Top five confusions

<table>
<thead>
<tr>
<th>class 1 / class 2</th>
<th>class 1 misclassified as class 2</th>
<th>class 2 misclassified as class 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ketch / schooner</td>
<td>21.6</td>
<td>14.8</td>
</tr>
<tr>
<td>lotus / water lily</td>
<td>15.3</td>
<td>20.0</td>
</tr>
<tr>
<td>crocodile / crocodile head</td>
<td>10.5</td>
<td>10.0</td>
</tr>
<tr>
<td>crayfish / lobster</td>
<td>11.3</td>
<td>9.1</td>
</tr>
<tr>
<td>flamingo / ibis</td>
<td>9.5</td>
<td>10.4</td>
</tr>
</tbody>
</table>

Easiest and hardest classes

- minaret (97.6%)
- windsor chair (94.6%)
- joshua tree (87.9%)
- okapi (87.8%)
- cougar body (27.6%)
- beaver (27.5%)
- crocodile (25.0%)
- ant (25.0%)

• Sources of difficulty: lack of texture, camouflage, “thin” objects, highly deformable shape
PMK/SIFT Best Categories (1-5)

100% 100%
99.7% 99.1%
98.2%
PMK/SIFT Best Categories (6-10)

97.7%

97.4%

95.7%

95.3%

95.2%
PMK/SIFT 5 Worst Categories

- Ants: 7.7%
- Beavers: 11.2%
- Crabs: 11.5%
- Birds: 11.8%
- Anchors: 12.3%
schooner n.
A fore-and-aft rigged sailing vessel having at least two masts, with a foremast that is usually smaller than the other masts.

ketch n. Nautical
A two-masted fore-and-aft-rigged sailing vessel with a mizzenmast stepped aft of a taller mainmast but forward of the rudder.
PMK/SIFT Most Confused Category Pairs

lotus

water lily
PMK/SIFT Most Confused Category Pairs

gerenuk

kangaroo
PMK/SIFT Most Confused Category Pairs

nautilus

brain
Fast intersection kernel SVMs for Realtime Object Detection

Subhransu Maji
UC Berkeley

Joint work with:
Alex Berg (Columbia University & UC Berkeley)
and
Jitendra Malik (UC Berkeley)
Fast intersection kernel SVMs for Realtime Object Detection

- IK SVM is a (simple) generalization of a linear SVM
- Can be evaluated very efficiently (sublinear in #SV)
- Other kernels (including ) have a similar form
- Methods applicable to current most successful object recognition/detection strategies.

Maji, Berg & Malik, CVPR 2008
Detection: Is this an X?

Ask this question over and over again, varying position, scale, category, pose…
Speedups: hierarchical, early reject, feature sharing, cueing but same underlying question!
Detection: Is this an X?

Ask this question over and over again, varying position, scale, multiple categories… Speedups: hierarchical, early reject, feature sharing, but same underlying question!
Detection: Is this an X?

Boosted dec. trees, cascades
- Very fast evaluation
- Slow training (esp. multi-class)

Linear SVM
- Fast evaluation
- Fast training
- Need to find good features

Non-linear kernelized SVM
- Better class. acc. than linear
- Medium training
- Slow evaluation

Ask this question over and over again, varying position, scale, multiple categories…
Speedups: hierarchical, early reject, feature sharing, but same underlying question!
Detection: Is this an X?

Boosted dec. trees, cascades
- Very fast evaluation
- Slow training (esp. multi-class)

Linear SVM
- Fast evaluation
- Fast training
- Need to find good features

Non-linear kernelized SVM
- Better class. acc. than linear
- Medium training
- Slow evaluation

Ask this question over and over again, varying position, scale, multiple categories…

Speedups: hierarchical, early reject, feature sharing, but same underlying question!
Outline

- What is Intersection Kernel SVM?
  - Brief Overview of Support Vector Machines
  - Multi-scale features based on Oriented Energy

- Algorithms
  - Algorithm to make classification fast (exact)
  - Algorithm to make classification very fast (approximate)

- Experimental Results

- Summary of where this matters
Outline

- What is Intersection Kernel SVM?
  - Brief Overview of Support Vector Machines
  - Multi-scale features based on Oriented Energy
- Algorithms
  - Algorithm to make classification fast (exact)
  - Algorithm to make classification very fast (approximate)
- Experimental Results
- Summary of where this matters
Support Vector Machines

Examples are:

\[(x_1, \ldots, x_n, y)\] with \(y \in \{-1, 1\}\)

\[\vec{w} \cdot \vec{x} + b = 0\]

\[\vec{w} \cdot \vec{x} + b = -1\]

\[\vec{w} \cdot \vec{x} + b = +1\]

\[f(\vec{x}) = \begin{cases} 
1 & \text{if } \vec{w} \cdot \vec{x} + b \geq 1 \\
-1 & \text{if } \vec{w} \cdot \vec{x} + b \leq -1 
\end{cases}\]

Margin = \[\frac{2}{||\vec{w}||^2}\]
Kernel Support Vector Machines

**Kernel Function**
- Inner Product in Hilbert Space
- Learn Non Linear Boundaries

$$K(x, y) = \Phi(x)^T \Phi(y)$$

**Gaussian Kernel**

$$K(x, y) = \exp(-\gamma \|x - y\|^2)$$

**Classification Function**

$$h(x) = \text{sign} \left( \sum_{i=1}^{n} \alpha_i k(x, x_i) + b \right)$$
Training Stage

(+ examples)  (- examples)

Feature Representation

Discriminative Classifier
Multiscale Oriented Energy feature

Concatenate orientation histograms for each orange region.

Differences from HOG:
-- Hierarchy of regions
-- Only performing L1 normalization once (at 16x16)
What is the Intersection Kernel?

Histogram Intersection kernel between histograms \( a, b \)

\[
K(a, b) = \sum_{i=1}^{n} \min(a_i, b_i) \quad \text{where} \quad a_i \geq 0 \quad \text{and} \quad b_i \geq 0
\]
What is the Intersection Kernel?

Histogram Intersection kernel between histograms $a, b$

$$K(a, b) = \sum_{i=1}^{n} \min(a_i, b_i) \quad a_i \geq 0 \quad b_i \geq 0$$

$K$ small $\rightarrow$ $a, b$ are different
$K$ large $\rightarrow$ $a, b$ are similar

Intro. by Swain and Ballard 1991 to compare color histograms.
Odone et al 2005 proved positive definiteness.
Can be used directly as a kernel for an SVM.
Compare to
Generalizations: Pyramid Match Kernel (Grauman et. al.), Spatial Pyramid Match Kernel (Lazebnik et.al)
Linear SVM, Kernelized SVM, IK SVM

Decision function is
\[ \text{sign}(h(x)) \]

Linear:
\[ h(x) = w'x + b = \sum_{i=1}^{\#\text{dim}} w_i x_i + b \]

Non-linear Using Kernel
\[ h(x) = \sum_{j=1}^{\#\text{sv}} \alpha^j K(x, x^j) + b \]

Histogram Intersection Kernel
\[ = \sum_{j=1}^{\#\text{sv}} \left( \alpha^j \sum_{i=1}^{\#\text{dim}} \min(x_i, x^j_i) \right) + b \]
Kernelized SVMs slow to evaluate

Decision function is

$$\text{sign} \left( \sum_{j=1}^{\#sv} \alpha^j K(x, x^j) + b \right)$$

where:

- Feature vector to evaluate
- Sum over all support vectors
- Kernel Evaluation
- Feature corresponding to a support vector $l$

Arbitrary Kernel

$$h(x) = \sum_{j=1}^{\#sv} \alpha^j K(x, x^j) + b$$

Histogram Intersection Kernel

$$h(x) = \sum_{j=1}^{\#sv} \left( \alpha^j \sum_{i=1}^{\#dim} \min(x_i, x^j_i) \right) + b$$

SVM with Kernel Cost:

- # Support Vectors $\times$ Cost of kernel comp.

IKSVM Cost:

- # Support Vectors $\times$ # feature dimensions
Algorithm 1

Decision function is 

\[ h(x) = \sum_{j=1}^{\#sv} \alpha^j \left( \sum_{i=1}^{\#dim} \min(x_i, x_i^j) \right) + b \]

\[ = \sum_{i=1}^{\#dim} \left( \sum_{j=1}^{\#sv} \alpha^j \min(x_i, x_i^j) \right) + b \]

\[ = \sum_{i=1}^{\#dim} h_i(x_i) \]

Importantly, just sort the support vector values in each coordinate, and pre-compute.

\[ h_i(x_i) = \sum_{j=1}^{\#sv} \alpha^j \min(x_i, x_i^j) + b \]

\[ = \sum_{x_i^j < x_i} \alpha^j x_i^j + \left( \sum_{x_i^j \geq x_i} \alpha^j \right) x_i \]

To evaluate, find position of \( x_i^j \) in the sorted support vector values \( x_i^j \) (cost: \( \log \#sv \)) look up values, multiply & add.
Algorithm 1

Decision function is \( \text{sign}(h(x)) \) where:

\[
h(x) = \sum_{j=1}^{\#sv} \left( \sum_{i=1}^{\#dim} \min(x_i, x_i^j) \right) + b
\]

\[
= \sum_{i=1}^{\#dim} \left( \sum_{j=1}^{\#sv} \alpha^j \min(x_i, x_i^j) \right) + b
\]

\[
= \sum_{i=1}^{\#dim} h_i(x_i)
\]

Just sort the support vector values in each coordinate, and pre-compute

\[
h_i(x_i) = \sum_{j=1}^{\#sv} \alpha^j \min(x_i, x_i^j) + b
\]

To evaluate, find position of \( x_i \) in the sorted support vector values \( x_i^j \) (cost: \( \log \#sv \)) look up values, multiply & add
Algorithm 2

Decision function is $\text{sign} \left( h(x) \right)$ where:

$$
  h(x) = \sum_{i=1}^{\text{#dim}} \left( \sum_{j=1}^{\text{#sv}} \alpha^j \min(x_i, x_i^j) \right) + b
$$

$$
  h_i(x_i) = \sum_{j=1}^{\text{#sv}} \alpha^j \min(x_i, x_i^j) + b
$$

For IK $h_i$ is piecewise linear, and quite smooth, blue plot. We can approximate with fewer uniformly spaced segments, red plot. Saves time & space!
Algorithm 2

Decision function is \( \text{sign}(h(x)) \) where:

\[
h(x) = \sum_{i=1}^{\text{#dim}} \left( \sum_{j=1}^{\text{#sv}} \alpha_j \min(x_i, x_i^j) \right) + b
\]

\[
h_i(x_i) = \sum_{j=1}^{\text{#sv}} \alpha_j \min(x_i, x_i^j) + b
\]

For IK \( h_i \) is piecewise linear, and quite smooth, blue plot. We can approximate with fewer uniformly spaced segments, red plot. Saves time & space!
Toy Example: accuracy/runtime vs. #bins

- Runtime independent of #bins (on left)
- Accuracy improves with #bins (on right)
Toy Example: accuracy/runtime vs. #sup vec

- Runtime independent of #sup vec! (for approximate)
- 2-3 orders of magnitude faster than LibSVM.
- Runtime memory requirement independent of #sup vec!
Results - INRIA Pedestrian Dataset

<table>
<thead>
<tr>
<th>Classification Method</th>
<th>Detection Rate (2 FPPI)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear SVM</td>
<td>43.12 %</td>
<td>-</td>
</tr>
<tr>
<td>IKSVM (binary search)</td>
<td>86.59 %</td>
<td>473×</td>
</tr>
<tr>
<td>IKSVM (piecewise linear)</td>
<td>86.59 %</td>
<td>2594×</td>
</tr>
<tr>
<td>IKSVM (piecewise constant)</td>
<td>86.59 %</td>
<td>3098×</td>
</tr>
<tr>
<td>Dalal &amp; Triggs [8]</td>
<td>79.63 %</td>
<td>-</td>
</tr>
<tr>
<td>Dalal &amp; Triggs* [8]</td>
<td>82.51 %</td>
<td>-</td>
</tr>
</tbody>
</table>

Fig. 1. Detection Rate at 2 FPPI on INRIA Person dataset for various methods. We use 300 bins to approximate both the piecewise linear and piecewise constant classifiers. The final classifier has 6416 support vectors and we get about 473× speedup using the binary search, 2594× using the piecewise linear approximation and 3098× speedup using piecewise constant approximation, without any loss in accuracy over the exact method. The performance using linear SVM is on the same features is significantly worse. We also compare our method with the state of the art Dalal & Triggs detector which uses a linear SVM with a different set of features. All the detectors are run densely with a stride of 8px along the width and height and a scalaratio of $2^{1/8}$ (*scalaratio of 1.05 ~ $2^{1/10}$).

- Outperforms linear significantly using pHOG features.
- About 3-4x slower than linear SVM. Most time spent on computing features anyway.
- IKSVM on HOG beats linear on HOG (not shown in the table)
Errors
Fig. 3. Accuracy on Daimler Chrysler Pedestrians dataset for various methods. We use 100 bins to approximate both the piecewise linear and piecewise constant classifiers. The final classifier has 5140±392 support vectors and we get about 485× speedup using the binary search, 2253× using the piecewise linear approximation and 3100× speedup using piecewise constant approximation, without any significant loss in accuracy over the exact method. The performance using linear SVM is on the same features is significantly worse.

Fig. 2. Classification Accuracy of various methods on Caltech-101 dataset. Accuracy is reported using 15(left) and 30(right) training examples per category and tested on 50 images averaged over 5 random splits. 60 bins are used to approximate the classifier for both the piecewise linear and piecewise constant approximation. The accuracy of the approximations is similar (within the variance) of the exact method (binary search IKSVVM) while being significantly better than a linear SVM.
Results - Single Scale UIUC Cars

<table>
<thead>
<tr>
<th>Classification Method</th>
<th>Performance (%)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear SVM</td>
<td>89.8</td>
<td>-</td>
</tr>
<tr>
<td>IKSVM (binary search)</td>
<td>98.5</td>
<td>23×</td>
</tr>
<tr>
<td>IKSVM (piecewise linear)</td>
<td>98.5</td>
<td>65×</td>
</tr>
<tr>
<td>IKSVM (piecewise constant)</td>
<td>98.5</td>
<td>83×</td>
</tr>
<tr>
<td>Agarwal &amp; Roth [1]</td>
<td>79.0</td>
<td>-</td>
</tr>
<tr>
<td>Garg et al. [12]</td>
<td>88.0</td>
<td>-</td>
</tr>
<tr>
<td>Fregus et al. [11]</td>
<td>88.5</td>
<td>-</td>
</tr>
<tr>
<td>ISM [19]</td>
<td>97.5</td>
<td>-</td>
</tr>
<tr>
<td>Mutch &amp; Lowe [20]</td>
<td>99.6</td>
<td>-</td>
</tr>
<tr>
<td>Lampert et al. [17]</td>
<td>98.5</td>
<td>-</td>
</tr>
</tbody>
</table>

Fig. 4. Performance at Equal Error Rate on UIUC Single Scale Cars for various methods. We use 50 bins to approximate both the piecewise linear and piecewise constant classifiers. The IKSVM classifier has 212 support vectors and we get an average of about 23× speedup using the binary search, 65× using the piecewise linear approximation and 83× speedup using piecewise constant approximation, without any loss in accuracy over the exact method. The performance using linear SVM is significantly worse.
Results – ETHZ Dataset

Dataset: Ferrari et al., ECCV 2006
255 images, over 5 classes
training = half of positive images for a class
+ same number from the other classes (1/4 from each)
testing = all other images
large scale changes; extensive clutter
Results – ETHZ Dataset

- Beats many current techniques without any changes to our features/classification framework.
- Shape is an important cue (use Pb instead of OE)
- Recall at 0.3 False Positive per Image (shown below)

<table>
<thead>
<tr>
<th>Method</th>
<th>Applelogo</th>
<th>Bottle</th>
<th>Giraffe</th>
<th>Mug</th>
<th>Swan</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAS*</td>
<td>65.0</td>
<td>89.3</td>
<td>72.3</td>
<td>80.6</td>
<td>64.7</td>
<td>76.7</td>
</tr>
<tr>
<td>Our</td>
<td>86.1</td>
<td>81.0</td>
<td>62.1</td>
<td>78.0</td>
<td>100</td>
<td>81.4</td>
</tr>
</tbody>
</table>

*Ferarri et.al, IEEE PAMI - 08
Other kernels allow similar trick

Decision function is \( \text{sign}(h(x)) \) where:

\[
\begin{align*}
\text{IKSVM} & \\
 h(x) &= \sum_{j=1}^{\#sv} \alpha^j \left( \sum_{i=1}^{\#dim} \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#dim} \left( \sum_{j=1}^{\#sv} \alpha^j \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#dim} h_i(x_i)
\end{align*}
\]

\( h_i \) are piece-wise linear, uniformly spaced
piece-wise linear approx. is fast.

\[
\begin{align*}
\text{SVM} & \\
 h(x) &= \sum_{j=1}^{\#sv} \alpha^j \left( \sum_{i=1}^{\#dim} \frac{(x_i - x_i^j)^2}{\frac{1}{2}(x_i + x_i^j)} \right) + b \\
&= \sum_{i=1}^{\#dim} \left( \sum_{j=1}^{\#sv} \alpha^j \frac{(x_i - x_i^j)^2}{\frac{1}{2}(x_i + x_i^j)} \right) + b \\
&= \sum_{i=1}^{\#dim} h_i(x_i)
\end{align*}
\]

\( h_i \) not piece-wise linear, but we can still use an approximation for fast evaluation.
Results outside computer vision

Accuracy of IK vs Linear on Text classification

<table>
<thead>
<tr>
<th>Classification Method</th>
<th>R8</th>
<th>R52</th>
<th>20Ng</th>
<th>Cade12</th>
<th>WebKb</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (Linear Kernel)</td>
<td>0.9666(1)</td>
<td>0.9322(1)</td>
<td>0.8155(0.04)</td>
<td>0.5650(0.05)</td>
<td>0.8796(0.04)</td>
</tr>
<tr>
<td>SVM (Intersection Kernel)</td>
<td>0.9693(1)</td>
<td>0.9326(0.8)</td>
<td>0.8115(0.05)</td>
<td>0.5777(0.10)</td>
<td>0.9105(0.04)</td>
</tr>
</tbody>
</table>

Error rate of directly
+ iksvm (blue)
+ best kernel (green)
+ linear (red)
on SVM benchmark datasets
Conclusions

- Exact evaluation in $O(\log \#SV)$, approx in $O(1)$ (same as linear!)
- Runtime for approximate is $O(1)$ (same as linear!)
- Significantly outperforms linear on variety of vision/non vision datasets
- Technique applies to any additive kernel (e.g. pyramid match kernel, spatial pyramid match kernel, $-\chi^2$, etc)
- Represents some of the best Caltech 256, Pascal VOC 2007 methods.
- Training time is much worse compared to linear (Dual coordinate descent, PEGASOS)
- Inside news! Train Additive Kernel SVMs quickly using online stochastic gradient descent.
- Trains IKSVM based INRIA pedestrian detector ~50K feats of 4K dim in 100s. (compared to 3-4hours using LibSVM).
Discriminatively Trained Mixtures of Deformable Part Models

Pedro Felzenszwalb and Ross Girshick
University of Chicago

David McAllester
Toyota Technological Institute at Chicago

Deva Ramanan
UC Irvine

http://www.cs.uchicago.edu/~pff/latent
PASCAL Challenge

• ~10,000 images, with ~25,000 target objects.
  - Objects from 20 categories (person, car, bicycle, cow, table...).
  - Objects are annotated with labeled bounding boxes.
Model Overview

- Mixture of deformable part models (pictorial structures)
- Each component has global template + deformable parts
- Fully trained from bounding boxes alone
2 component bicycle model

- **Root filters**
  - Coarse resolution
- **Part filters**
  - Finer resolution
- **Deformation models**
Object Hypothesis

Score of filter is dot product of filter with HOG features underneath it

Score of object hypothesis is sum of filter scores minus deformation costs

Multiscale model captures features at two resolutions
Model

\[ f_w(x) = w \cdot \Phi(x) \]

\[ f_w(x) = \max_z w \cdot \Phi(x, z) \]

\( Z = \text{vector of part offsets} \)

\( \Phi(x, z) = \text{vector of HOG features (from root filter & appropriate part sub-windows) and part offsets} \)
Latent SVM

\[ f_w(x) = \max_z w \cdot \Phi(x, z) \]

Linear in \( w \) if \( z \) is fixed

Training data: \((x_1, y_1), \ldots, (x_n, y_n)\) with \( y_i \in \{-1, 1\} \)

Learning: find \( w \) such that \( y_i f_w(x_i) > 0 \)

\[ w^* = \arg\min_w \lambda \|w\|^2 + \sum_{i=1}^{n} \max(0, 1 - y_i f_w(x_i)) \]

Regularization \hspace{2cm} Hinge loss
Latent SVM training

\[ w^* = \arg\min_w \lambda \|w\|^2 + \sum_{i=1}^{n} \max(0, 1 - y_i f_w(x_i)) \]

- Non-convex optimization
- Huge number of negative examples
- Convex if we fix \( z \) for positive examples
- Optimization:
  - Initialize \( w \) and iterate:
    - Pick best \( z \) for each positive example
    - Optimize \( w \) via gradient descent with data mining
Initializing $\mathcal{W}$

- For $k$ component mixture model:
- Split examples into $k$ sets based on bounding box aspect ratio
- Learn $k$ root filters using standard SVM
  - Training data: warped positive examples and random windows from negative images (Dalal & Triggs)
- Initialize parts by selecting patches from root filters
  - Subwindows with strong coefficients
  - Interpolate to get higher resolution filters
  - Initialize spatial model using fixed spring constants
Car model

- root filters
  - coarse resolution
- part filters
  - finer resolution
- deformation models
Person model

- Root filters
- Coarse resolution
- Part filters
- Finer resolution
- Deformation models
Bottle model

root filters
coarse resolution

part filters
finer resolution

deformation models
Dalal & Triggs:
- Histogram gradient orientations in 8x8 pixel blocks (9 bins)
- Normalize with respect to 4 different neighborhoods and truncate
- 9 orientations * 4 normalizations = 36 features per block

PCA gives ~10 features that capture all information
- Fewer parameters, speeds up convolution, but costly projection at runtime

Analytic projection: spans PCA subspace and easy to compute
- 9 orientations + 4 normalizations = 13 features

We also use 2*9 contrast sensitive features for 31 features total
Bounding box prediction

- predict \((x_1, y_1)\) and \((x_2, y_2)\) from part locations
- linear function trained using least-squares regression
Context rescoring

• Rescore a detection using “context” defined by all detections

• Let $v_i$ be the max score of detector for class $i$ in the image

• Let $s$ be the score of a particular detection

• Let $(x_1, y_1), (x_2, y_2)$ be normalized bounding box coordinates

• $f = (s, x_1, y_1, x_2, y_2, v_1, v_2... , v_{20})$

• Train class specific classifier
  - $f$ is positive example if true positive detection
  - $f$ is negative example if false positive detection
Bicycle detection
More bicycles

False positives
Car
Code

Source code for the system and models trained on PASCAL 2006, 2007 and 2008 data are available here:

http://www.cs.uchicago.edu/~pff/latent
Today: Advanced kernels

- SVM-BOW
- Pyramid and spatial-pyramid match
- Fast IK
- Latent-part SVM models
Correspondence and Pyramid-based techniques

Discriminative approaches (SVM, HCRF)

• Classic SVM on “bags of features”:

• ISM + SVM + Local Kernels:
  M. Fritz; B. Leibe; B. Caputo; B. Schiele: Integrating Representative and Discriminant Models for Object Category Detection, ICCV’05, Beijing, China, 2005

• Local SVM:

• “Latent” SVM with deformable parts:

• Hidden Conditional Random Fields: