Classification using intersection kernel SVMs is efficient

Jitendra Malik
UC Berkeley

Joint work with Subhransu Maji and Alex Berg

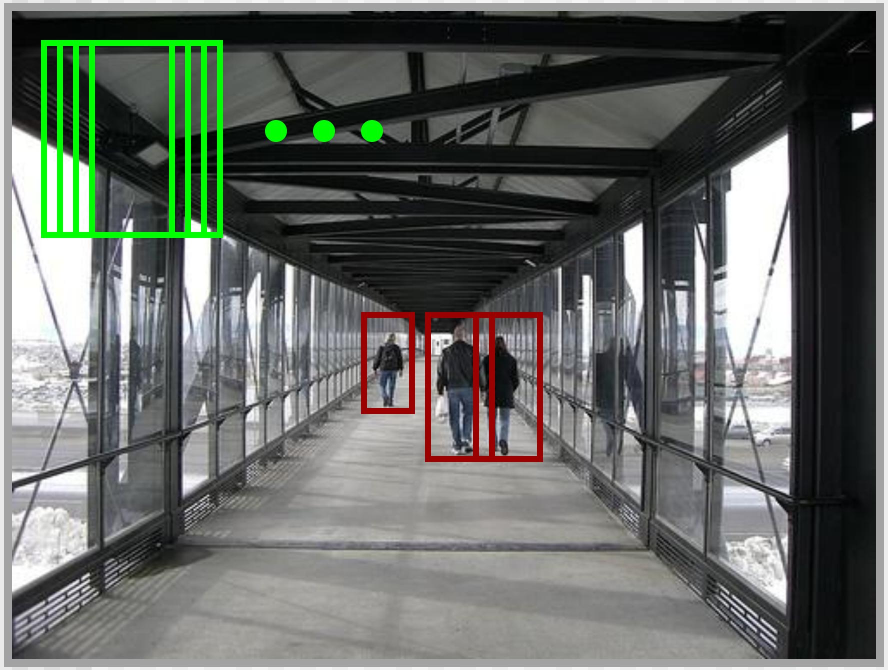
Fast intersection kernel SVMs
and other generalizations of linear SVMs

- IKSVM is a (simple) generalization of a linear SVM
- Can be evaluated very efficiently
- Other kernels (including $-\chi^2$) have a similar form
- Novel features based on pyramid of oriented energy.
- Methods applicable to current most successful object recognition/detection strategies.

Detection: Is this an X?

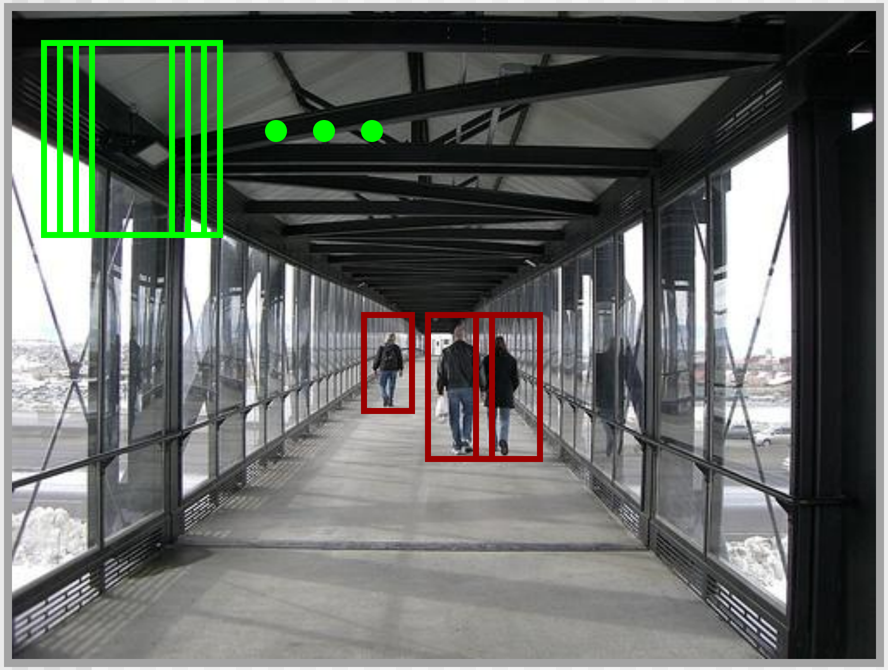
Ask this question over and over again,
varying position, scale, multiple categories…
Speedups: hierarchical, early reject, feature sharing, cueing
but same underlying question!

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?



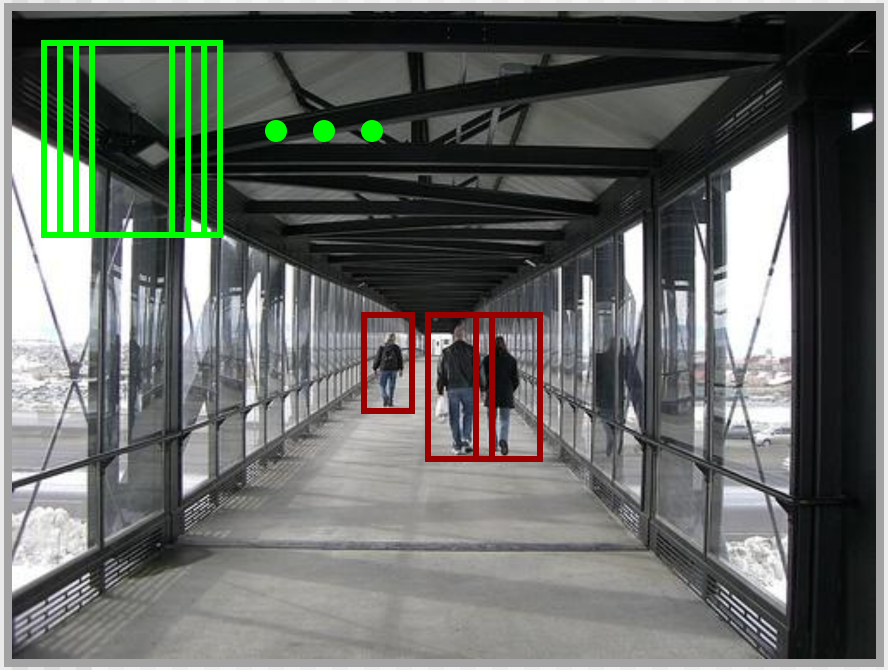
Boosted dec. trees, cascades
  + Very fast evaluation
  - Slow training (esp. multi-class)

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

This work

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!

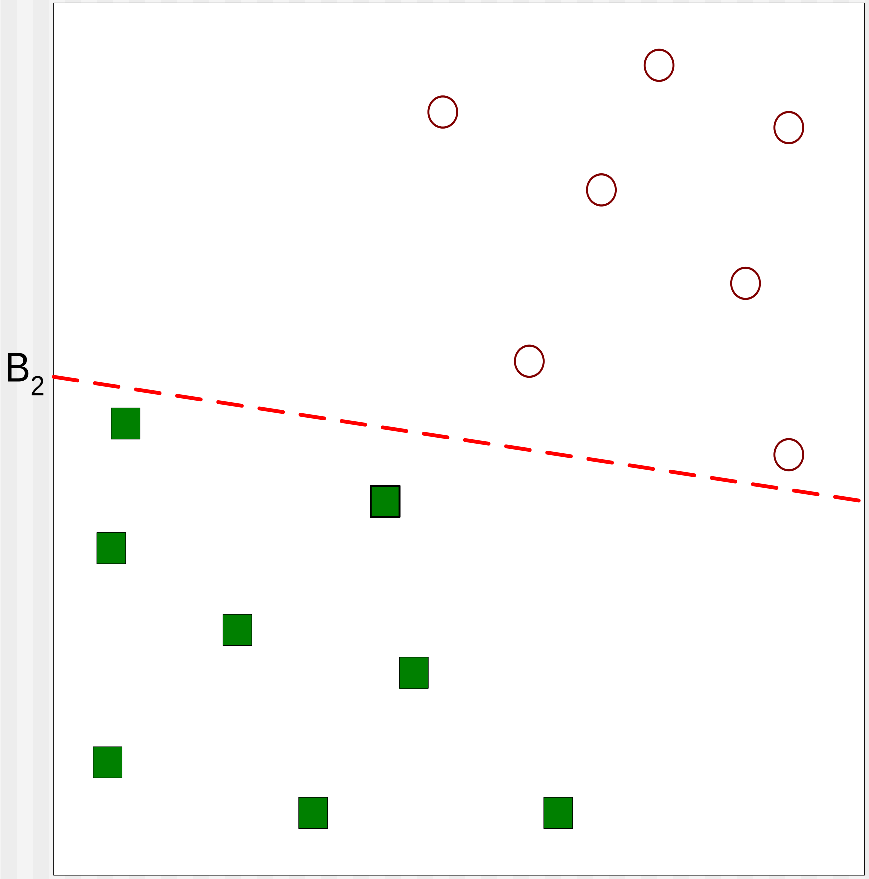
Outline

- What is Intersection Kernel SVM?
  - Trick to make it fast (exact)
  - Trick to make it very fast (approximate)
  - Why use it?
  - Multi-scale Features based on Oriented Energy
- Generalization of linear classifiers
  - Reinterpret the approximate IKSVM
  - Fast training
- Summary of where this matters

Outline

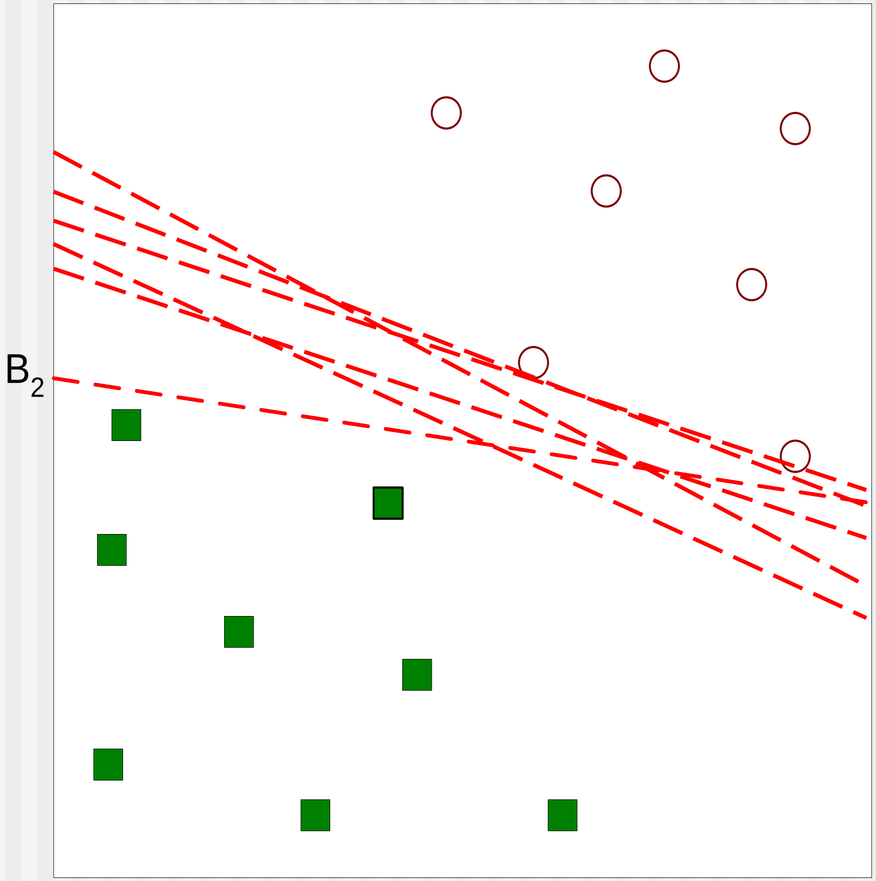
- What is Intersection Kernel SVM?
  - Trick to make it fast (exact)
  - Trick to make it very fast (approximate)
  - Why use it?
  - Multi-scale Features based on Oriented Energy
- Generalization of linear classifiers
  - Reinterpret the approximate IKSVM
  - Fast training
- Summary of where this matters

Support Vector Machines



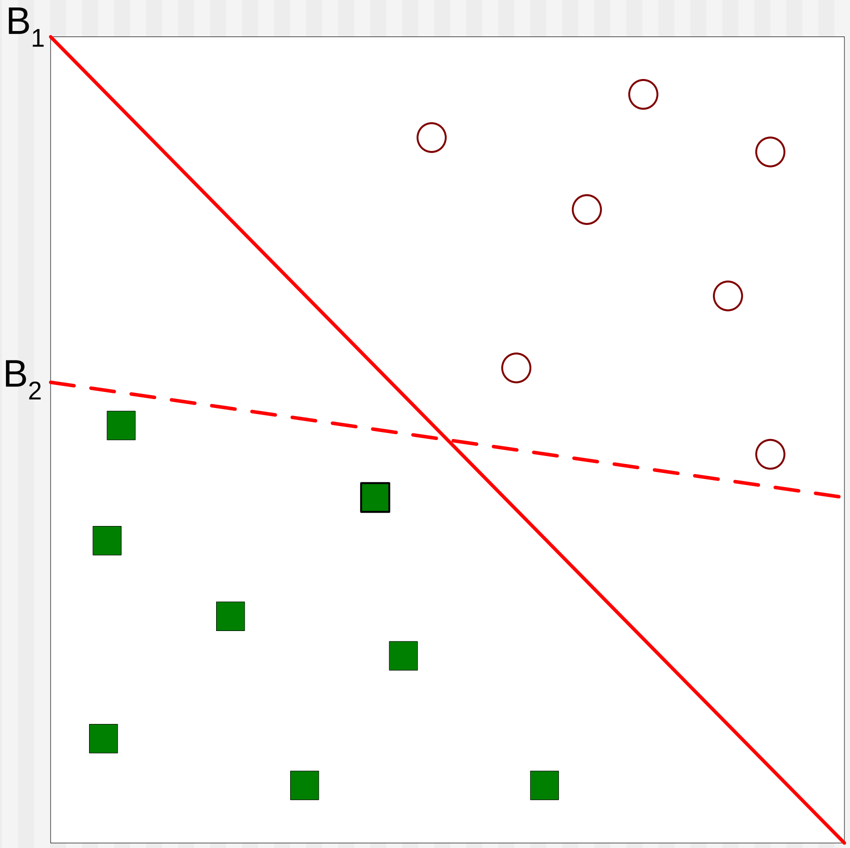
- Linear Separators (aka. Perceptrons)

Support Vector Machines



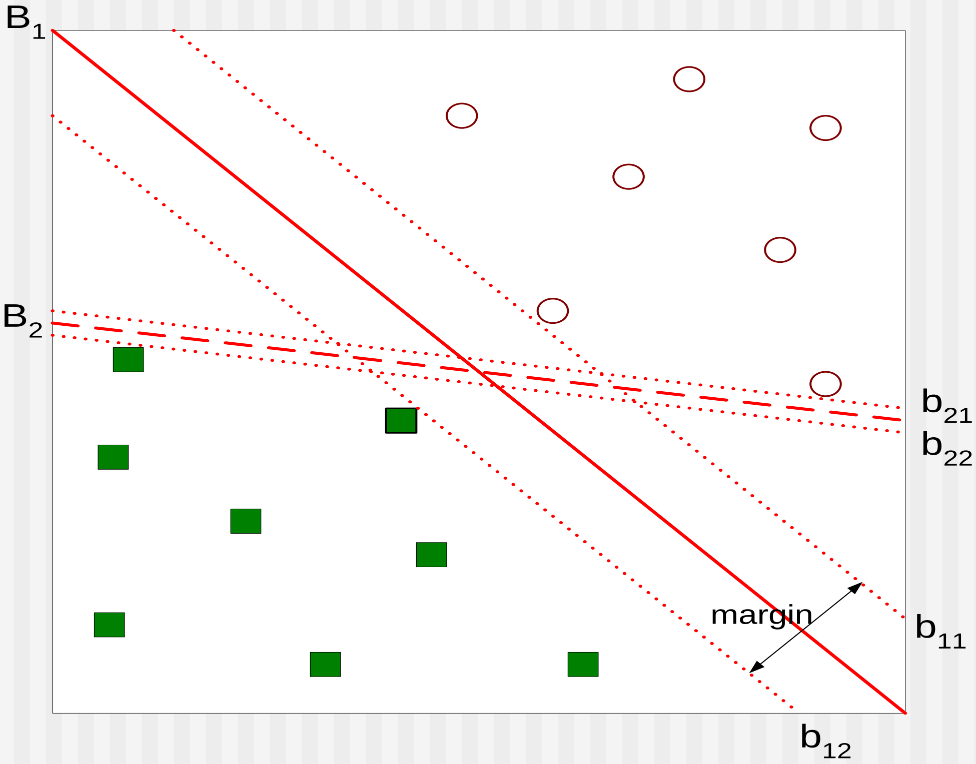
- Other possible solutions

Support Vector Machines



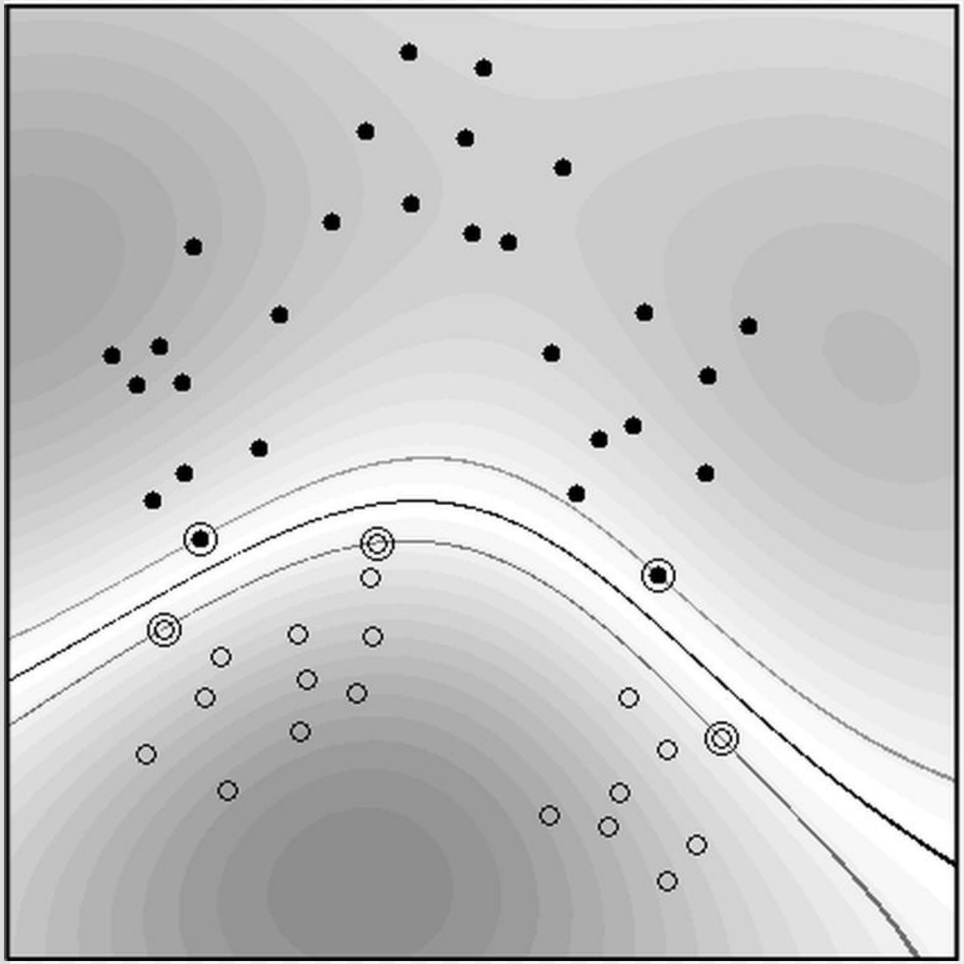
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



- Find hyperplane maximizes the margin => B1 is better than B2

Kernel Support Vector Machines



Kernel :

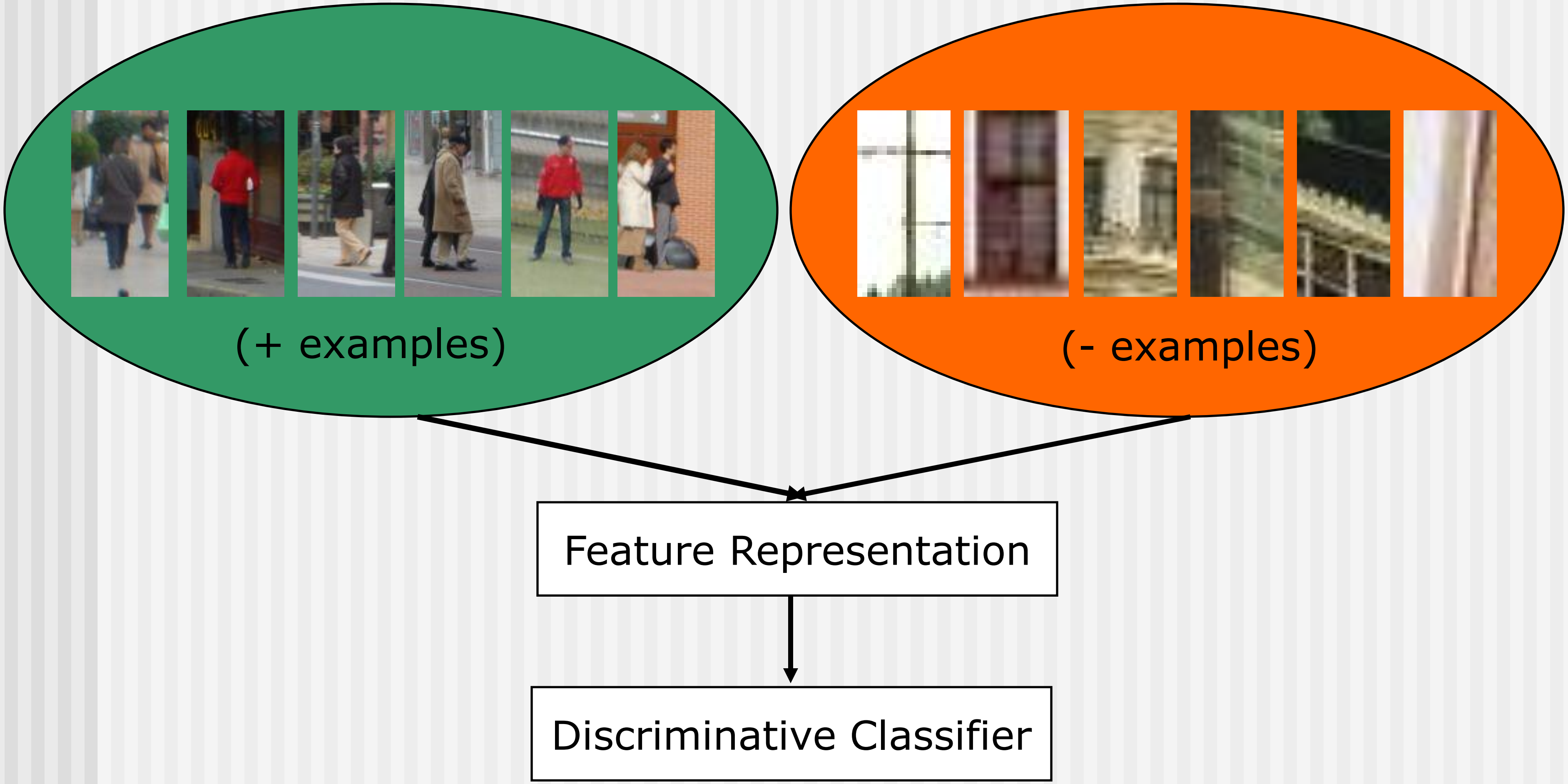
•Inner Product in Hilbert Space

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

•Can Learn Non Linear Boundaries

$$K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$$

Training Stage



Our Multiscale HOG-like feature

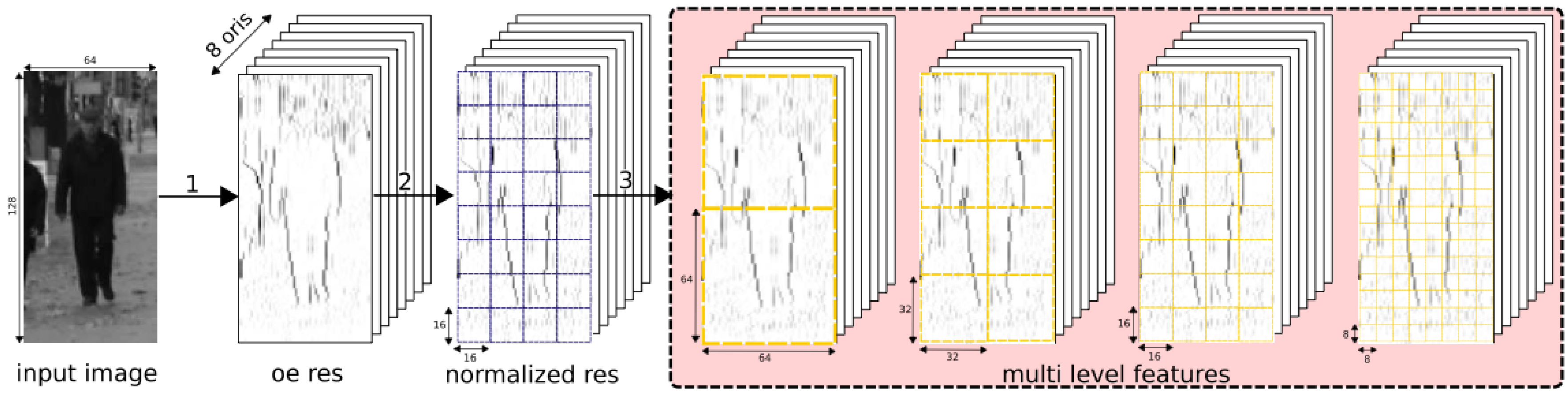


Figure 2. The three stage pipeline of the feature computation process. **(1)** The input grayscale image of size 64×128 is convolved with oriented filters ($\sigma = 1$) in 8 directions, to obtain oriented energy responses. **(2)** The responses are then L_1 normalized over all directions in each non overlapping 16×16 blocks independently to obtain normalized responses. **(3)** Multilevel features are then extracted by constructing histograms of oriented gradients by summing up the normalized response in each cell. The diagram depicts progressively smaller cell sizes from 64×64 to 8×8.

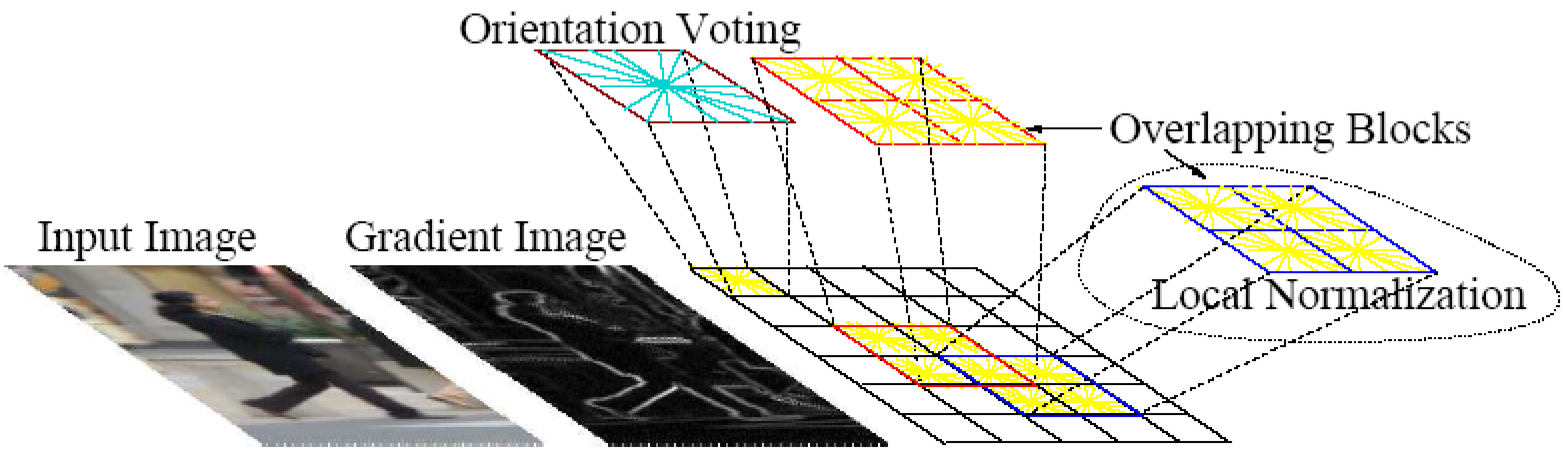
Concatenate orientation histograms for each orange region.

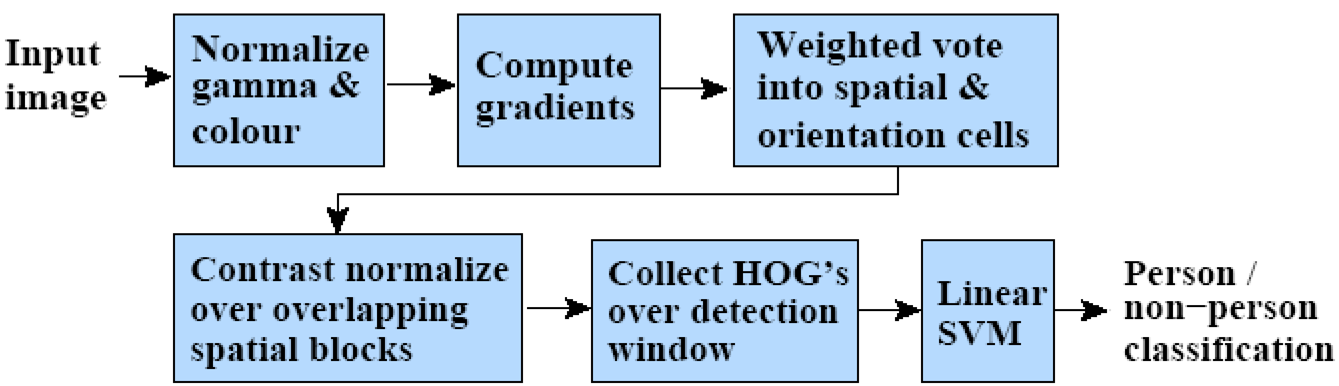
Differences from HOG:

-- Hierarchy of regions

-- Only performing L1 normalization once (at 16x16)

Comparison to HOG (Dalal & Triggs)





Comparison to HOG (Dalal & Triggs)

- Smaller Dimensional (1360 vs. 3780)
- Simple Implementation (Convolutions)
- Faster to compute

  + No non-local Normalization

  + No gaussian weighting

  + No color normalization

What is the Intersection Kernel?

Histogram Intersection kernel between histograms *a*, *b*

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

$$a_i \geq 0$$
$$b_i \geq 0$$

What is the Intersection Kernel?

Histogram Intersection kernel between histograms *a*, *b*

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

K small -> *a*, *b* are different
K large -> *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 $-\chi^2$

linear SVM, Kernelized SVM, IKSVM

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

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

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

Kernelized SVMs slow to evaluate

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

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

SVM with Kernel Cost:

Support Vectors x Cost of kernel comp.

IKSVM Cost:

Support Vectors x # feature dimensions

The Trick

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

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

$$
\begin{aligned}
h_i(x_i) &= \sum_{j=1}^{\#\text{sv}} \alpha^j \min(x_i, x_i^j) + b \\
&= \sum_{x_i^j < x_i} \alpha^j x_i^j + \left(\sum_{x_i^j \geq x_i} \alpha^j \right) x_i
\end{aligned}
$$

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

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

The Trick

~~#support vectors x #dimensions~~

log(#support vectors) x #dimensions

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

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

$$
\begin{aligned}
h_i(x_i) &= \sum_{j=1}^{\#\text{sv}} \alpha^j \min(x_i, x_i^j) + b \\
&= \sum_{x_i^j < x_i} \alpha^j x_i^j + \left(\sum_{x_i^j \geq x_i} \alpha^j \right) x_i
\end{aligned}
$$

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

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

The Trick 2

~~#support vectors x #dimensions~~

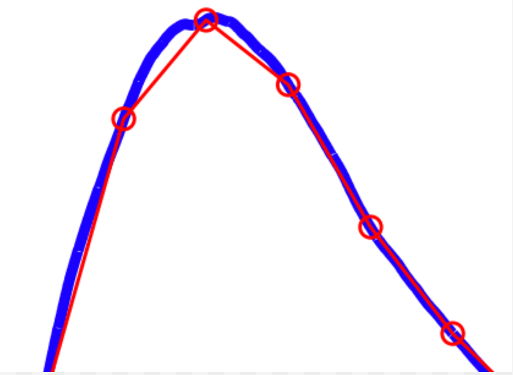
log(#support vectors) x #dimensions

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

$$
\begin{aligned}
h(x) &= \sum_{i=1}^{\#\mathrm{dim}} \left(\sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} h_i(x_i)
\end{aligned}
$$

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!

$$
\begin{aligned}
h_i(x_i) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) + b \\
&= \sum_{x_i^j < x_i} \alpha^j x_i^j + \left(\sum_{x_i^j \geq x_i} \alpha^j \right) x_i
\end{aligned}
$$



~~#support vectors x #dimensions~~

log(~~#support vectors) x #dimensions~~

constant x #dimensions

The Trick 2

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

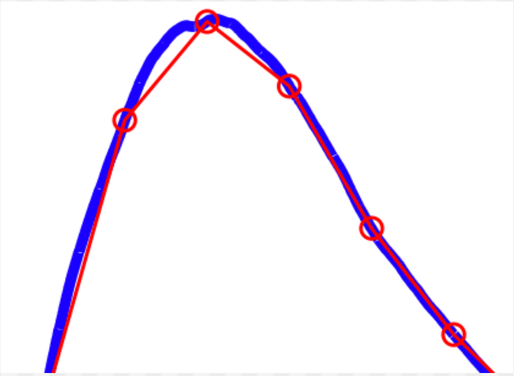
$$h(x) = \sum_{i=1}^{\#\mathrm{dim}} \left(\sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) \right) + b$$

$$= \sum_{i=1}^{\#\mathrm{dim}} h_i(x_i)$$

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!

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

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



Timing Results

reduced memory!

Time to evaluate 10,000 feature vectors

Dataset	Model parameters		SVM kernel type		fast IKSVMs		
	#SVs	#features	linear	intersection	binary search	piecewise-const	piecewise-lin
INRIA Ped	3363	1360	0.07±0.00	659.1±1.92	2.57±0.03	0.34±0.01	0.43±0.01
DC Ped	5474±395	656	0.03±0.00	459.1±31.3	1.43±0.02	0.18±0.01	0.22±0.00
Caltech 101	175±46	1360	0.07±0.01	24.77±1.22	1.63±0.12	0.33±0.03	0.46±0.03

Linear SVM with our multi-scale
Version of HOG features
has worse classification perf.
than Dalal & Triggs.

IKSVM with our multi-scale
version of HOG features
beats Dalal & Triggs. Also
for Daimler Chrysler data.
Current Best on these datasets.

Distribution of support vector values and h_i

Distribution of x_i^j

$h_i(x_i)$

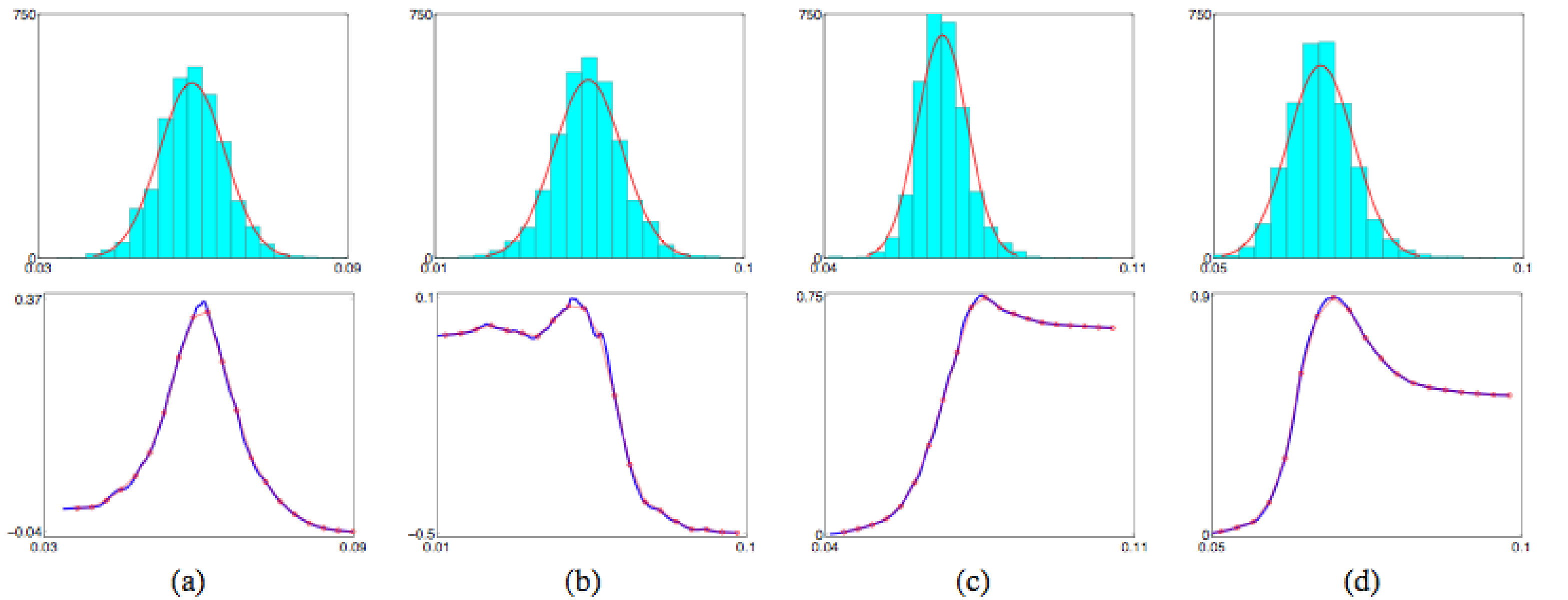
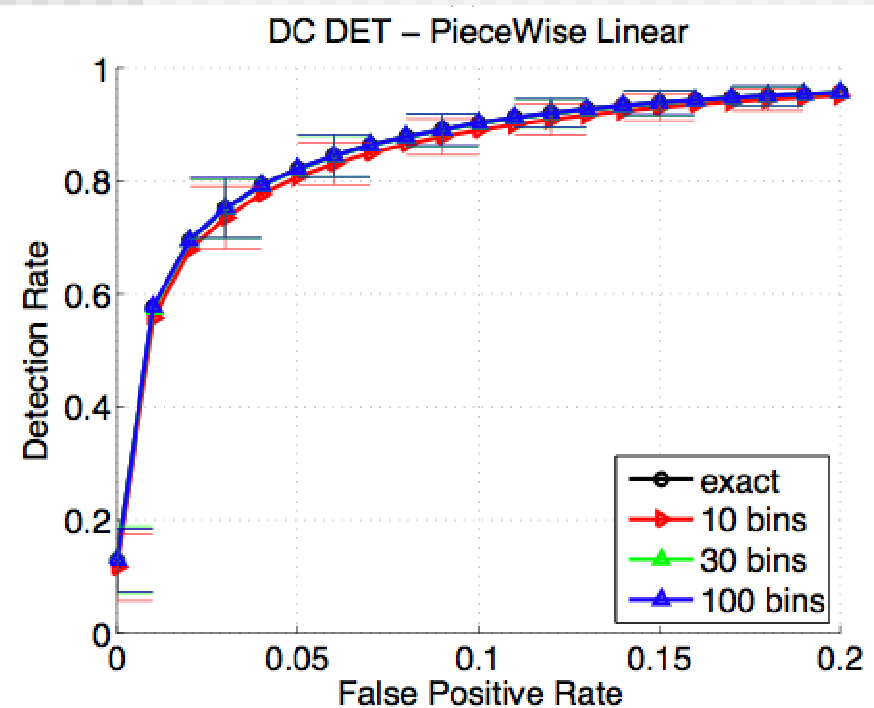


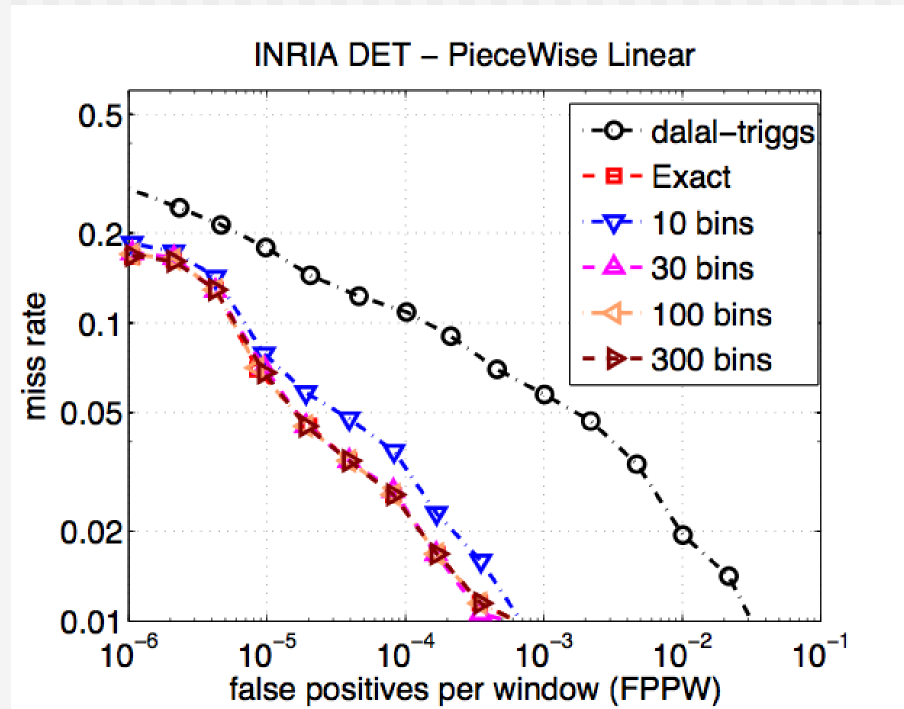
Figure 1. Each column (a-d) shows the distribution of the support vectors values along a dimension with a Gaussian fit (top) and the function $h_i(x)$ vs. x with a piecewise linear fit using 20 uniformly spaced points (bottom) of an IKSVM model trained on the INRIA dataset. Unlike the distribution of the training data which are heavy tailed, the values of the support vectors tend to be clustered.

Best Performance on Pedestrian Detection, Improve on Linear for Many Tasks

Daimler Chrysler Pedestrians



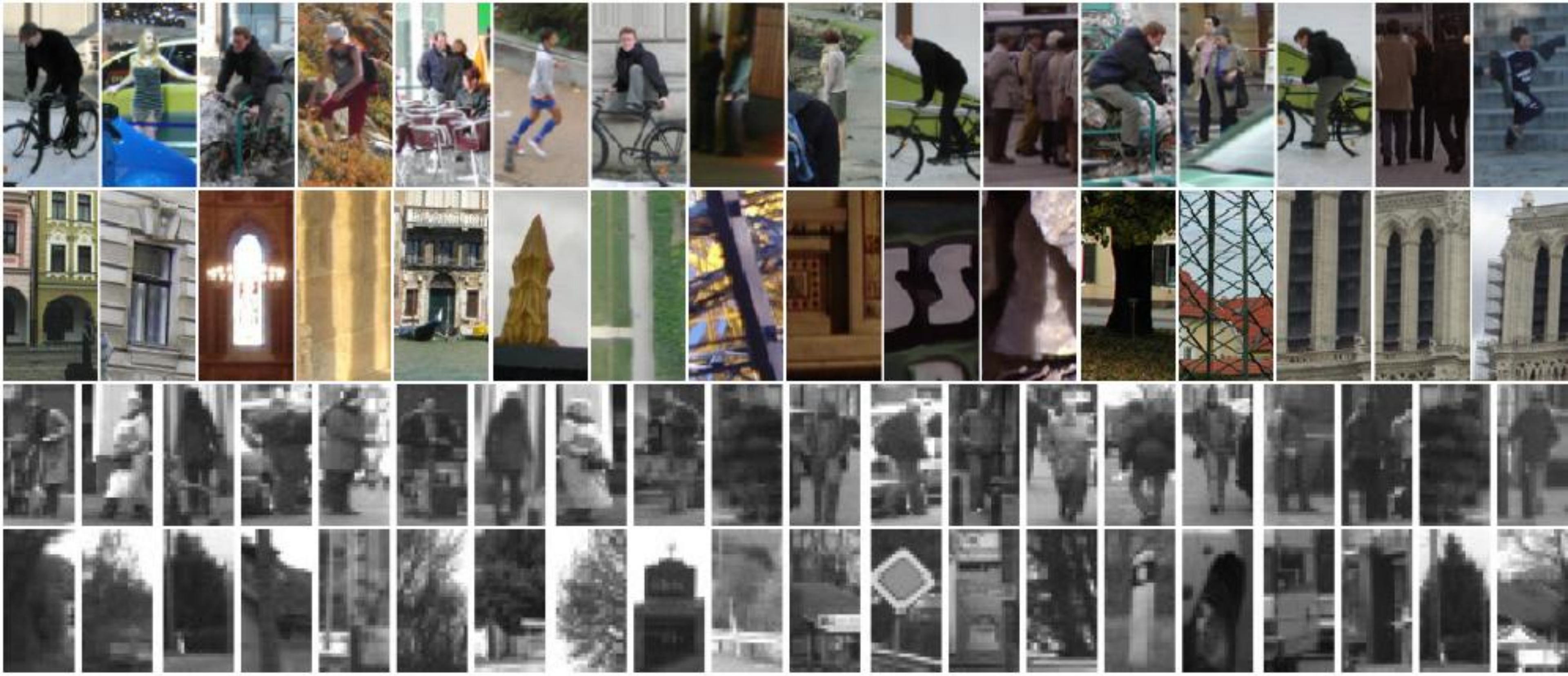
INRIA Pedestrians



Caltech 101 with “simple features” Linear SVM 40% correct

IKSVM 52% correct

Classification Errors



Results – ETHZ Dataset

Dataset: Ferrari et al., ECCV 2006

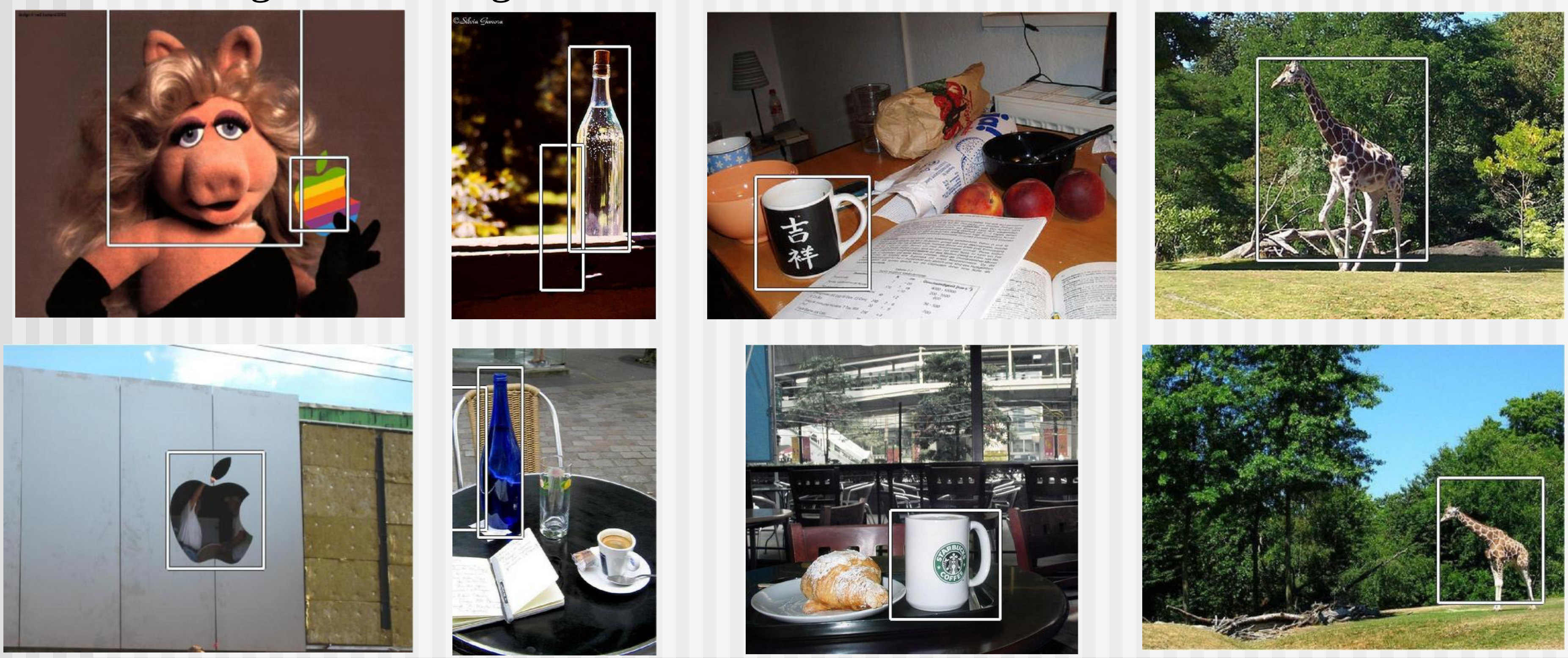
255 images, over 5 classes

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.
- Recall at 0.3 False Positive per Image
- Shape is an important cue (use Pb instead of OE)

Method	Applelogo	Bottle	Giraffe	Mug	Swan	Avg
PAS*	65.0	89.3	72.3	80.6	64.7	76.7
Our	86.1	81.0	62.1	78.0	100	81.4

*Ferarri *et.al,* IEEE PAMI - 08

Other kernels allow similar trick

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

IKSVM

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

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

$-\chi^2$ SVM

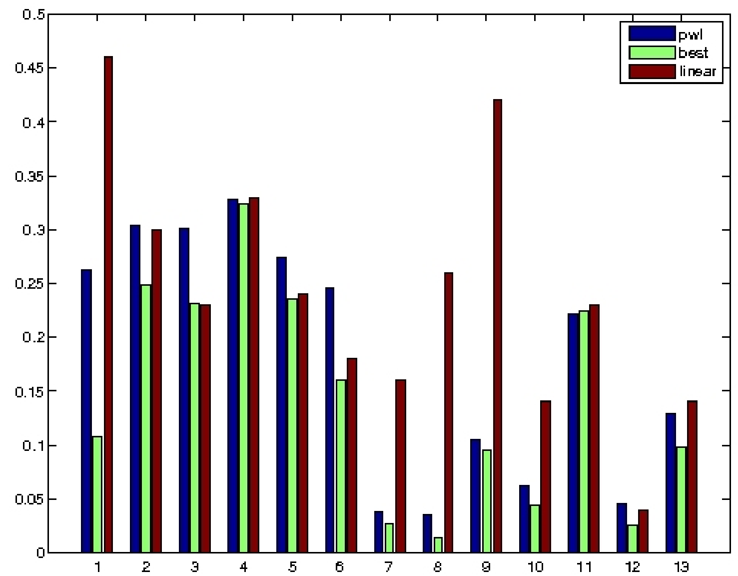
$$h(x) = \sum_{j=1}^{\#\text{sv}} \alpha^j \left(\sum_{i=1}^{\#\text{dim}} \frac{(x_i - x_i^j)^2}{\frac{1}{2}(x_i + x_i^j)} \right) + b$$
$$= \sum_{i=1}^{\#\text{dim}} \left(\sum_{j=1}^{\#\text{sv}} \alpha^j \frac{(x_i - x_i^j)^2}{\frac{1}{2}(x_i + x_i^j)} \right) + b$$
$$= \sum_{i=1}^{\#\text{dim}} h_i(x_i)$$

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

Accuracy Values					
Classification Method	**R8**	**R52**	**20Ng**	**Cade12**	**WebKb**
SVM (Linear Kernel)	0.9666(1)	0.9322(1)	0.8155(0.04)	0.5650(0.05)	0.8796(0.04)
SVM (Intersection Kernel)	0.9693(1)	0.9326(0.8)	0.8115(0.05)	0.5777(0.10)	0.9105(0.04)

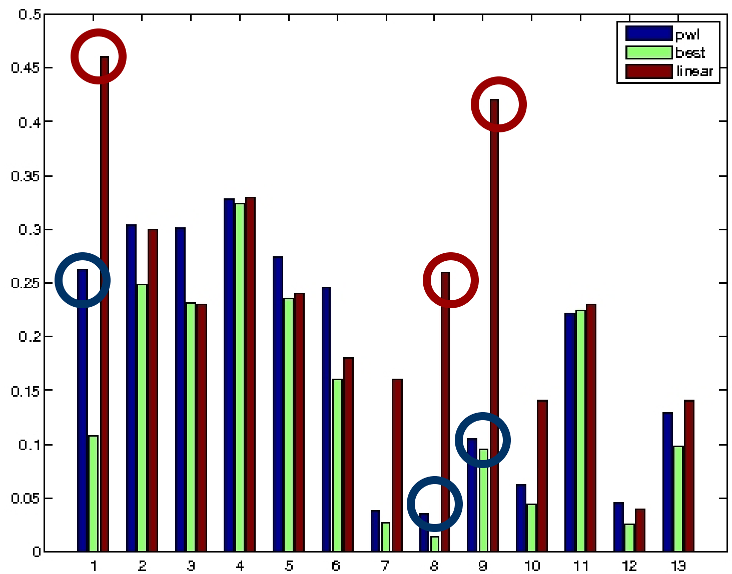


Error rate of directly
trained piecewise linear (blue)
best kernel (green)
and linear (red)
on SVM benchmark datasets

Results outside computer vision

Accuracy of IK vs Linear on Text classification

Accuracy Values					
Classification Method	**R8**	**R52**	**20Ng**	**Cade12**	**WebKb**
SVM (Linear Kernel)	0.9666(1)	0.9322(1)	0.8155(0.04)	0.5650(0.05)	0.8796(0.04)
SVM (Intersection Kernel)	0.9693(1)	0.9326(0.8)	0.8115(0.05)	0.5777(0.10)	0.9105(0.04)

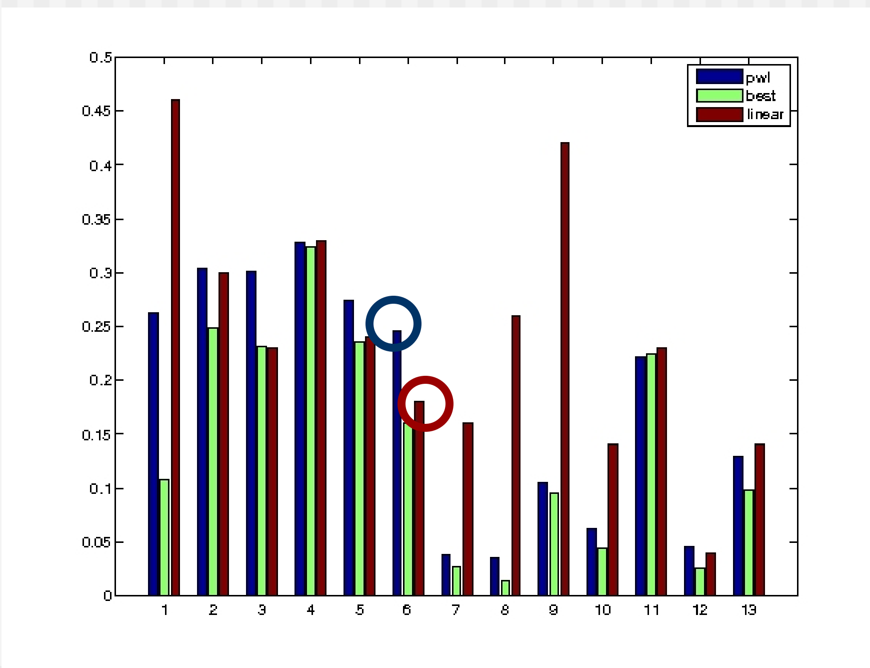


Error rate of directly
trained piecewise linear (blue)
best kernel (green)
and linear (red)
on SVM benchmark datasets

Results outside computer vision

Accuracy of IK vs Linear on Text classification

Accuracy Values					
Classification Method	**R8**	**R52**	**20Ng**	**Cade12**	**WebKb**
SVM (Linear Kernel)	0.9666(1)	0.9322(1)	0.8155(0.04)	0.5650(0.05)	0.8796(0.04)
SVM (Intersection Kernel)	0.9693(1)	0.9326(0.8)	0.8115(0.05)	0.5777(0.10)	0.9105(0.04)



Error rate of directly
trained piecewise linear (blue)
best kernel (green)
and linear (red)
on SVM benchmark datasets

Piecewise linear usually better,
Depending on amount of data
relative to the dimension and
regularization

Conclusions

- Exactly evaluate IKSVM in O(n log m) as opposed to O(nm)
  - Makes SV cascade or other ordering schemes irrelevant for intersection kernel
- Verified that IKSVM offers classification performance advantages over linear
- Approximate decision functions that decompose to a sum of functions for each coordinate (including Chi squared)
- Directly learn such classification functions (no SVM machinery)
- Generalized linear svm beats linear SVM in some applications often as good as more expensive RBF kernels
- Showed that relatively simple features with IKSVM beats Dalal & Triggs (linear SVM), leading to the state of the art in pedestrian detection.
- Applies to best Caltech 256, Pascal VOC 2007 methods.

Classification Using Intersection Kernel Support Vector Machines is efficient.
Subhransu Maji and Alexander C. Berg and Jitendra Malik.
Proceedings of CVPR 2008, Anchorage, Alaska, June 2008.

Software and more results available at
http://www.cs.berkeley.edu/~smaji/projects/fiksvm/