Classifiers and Detection

D.A. Forsyth

Classifiers

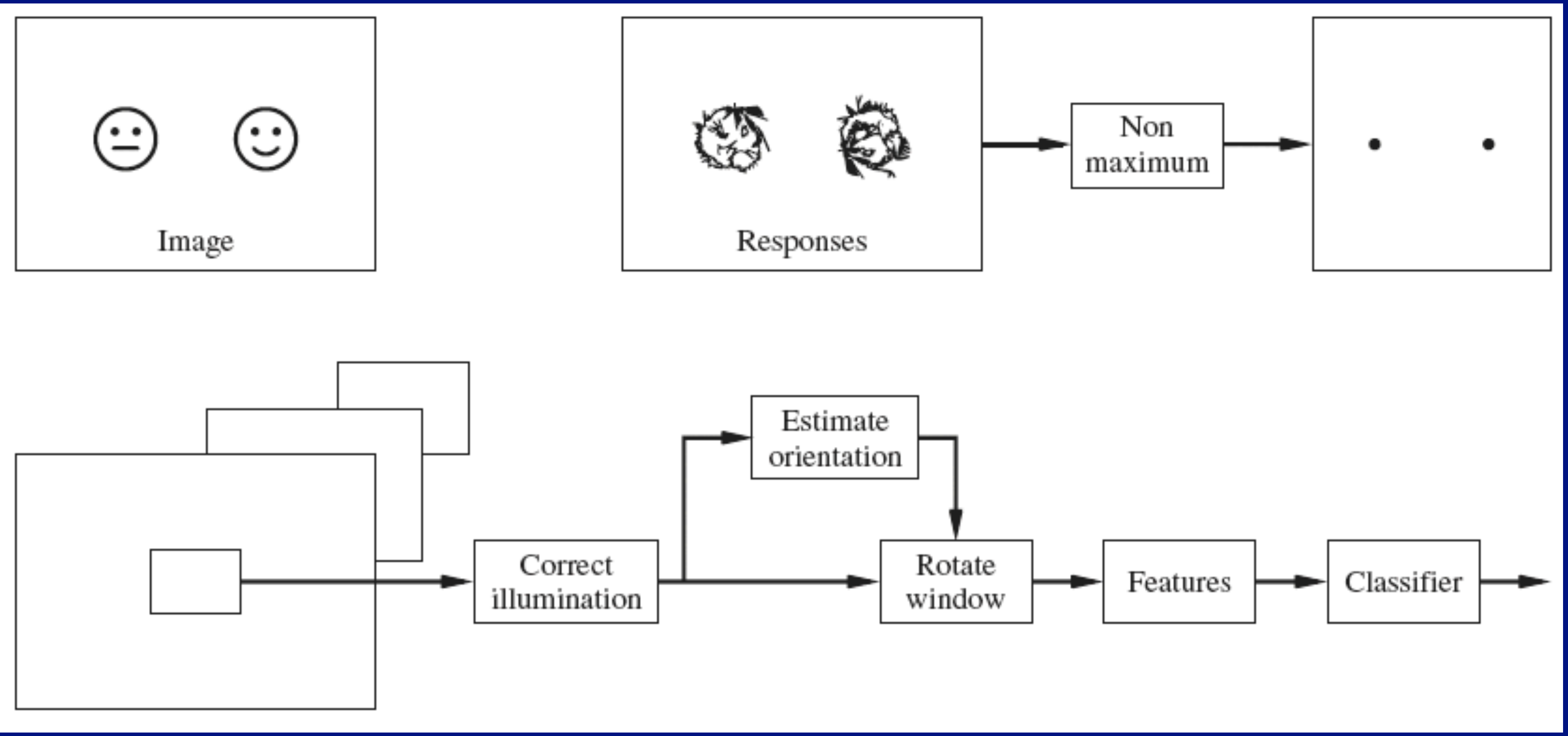
- Take a measurement x, predict a bit (yes/no; 1/-1; 1/0; etc)

Detection with a classifier

- Search
  - all windows
  - at relevant scales
- Prepare features
- Classify

- Issues
  - how to get only one response
  - speed
  - accuracy

Detection with a classifier



Non-maximum suppression

- Compute “strength of response”
  - SVM value
  - LR value
- threshold
  - small values are not faces
- find largest value (over location, scale)
  - suppress nearby values
  - repeat

Classifiers

- Take a measurement x, predict a bit (yes/no; 1/-1; 1/0; etc)
- Strategies:
  - non-parametric
    - nearest neighbor
  - probabilistic
    - histogram
    - logistic regression
  - decision boundary
    - SVM

Basic ideas in classifiers

- Loss
  - errors have a cost, and different types of error have different costs
    - this means each classifier has an associated risk
  - Total risk

$$R(s) = Pr\{1 \to 2|\text{using } s\}\, L(1 \to 2) + Pr\{2 \to 1|\text{using } s\}\, L(2 \to 1)$$

  - Bayes risk
    - smallest possible value of risk, over all classification strategies

Nearest neighbor classification

- Examples
  - (x_i, y_i)
    - here y is yes/no or -1/1 or 1/0 or....
  - training set
- Strategy
  - to label new example (test example)
    - find closest training example
    - report its label
- Advantage
  - in limit of very large number of training examples, risk is 2*bayes risk
- Issue
  - how do we find closest example?
  - what distance should we use?

k-nearest neighbors

- Strategy
  - to classify test example
    - find k-nearest neighbors of test point
    - vote (it’s a good idea to have k odd)
- Issues
  - how do we find nearest neighbors?
  - what distance should we use?

Nearest neighbors

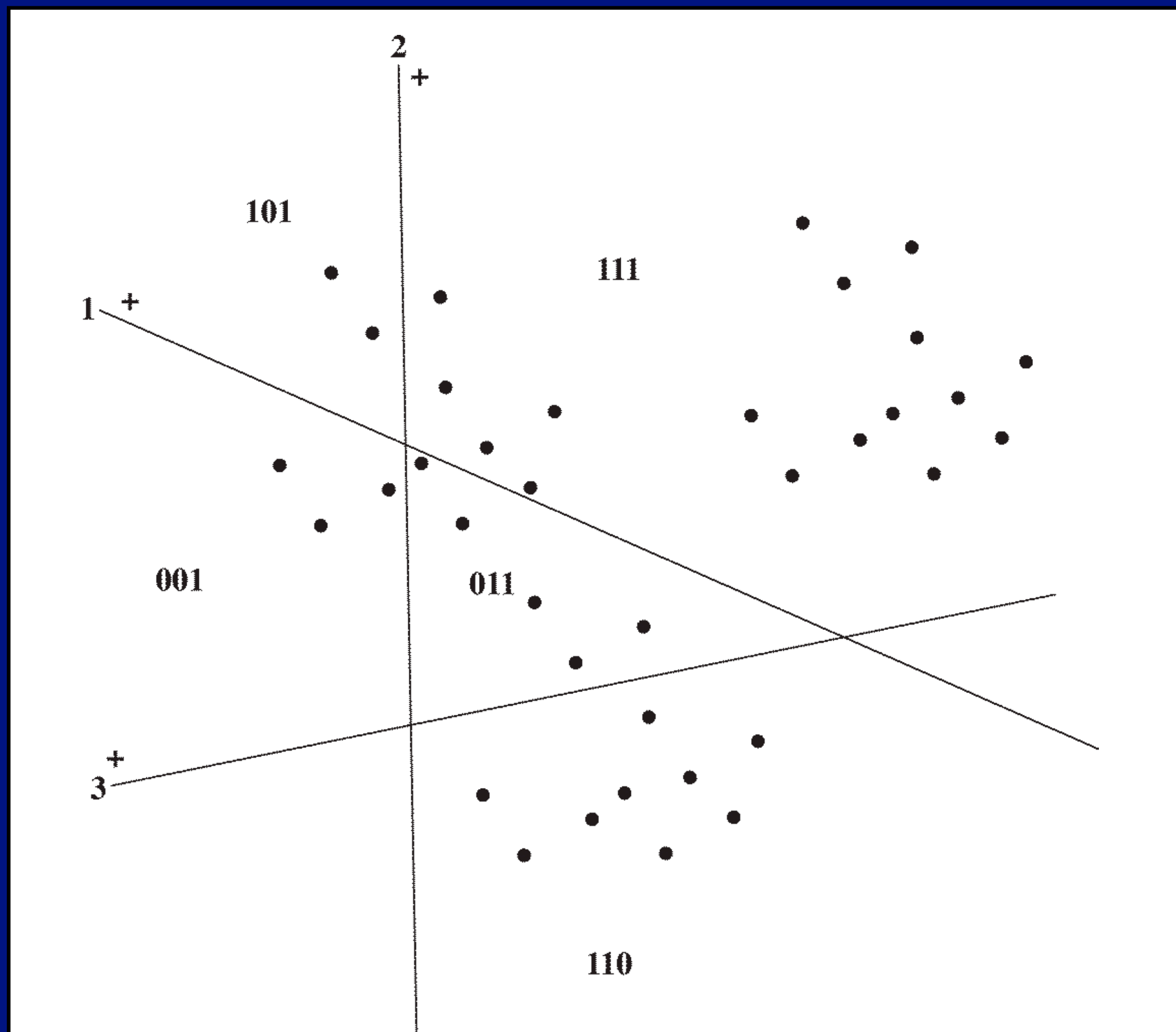
- Exact nearest neighbor in large dataset
  - linear search is very good
  - very hard to do better (surprising fact)
- Approximate nearest neighbor is easier
  - methods typically give probabilistic guarantees
  - good enough for our purposes
  - methods
    - locality sensitive hashing
    - k-d tree with best bin first

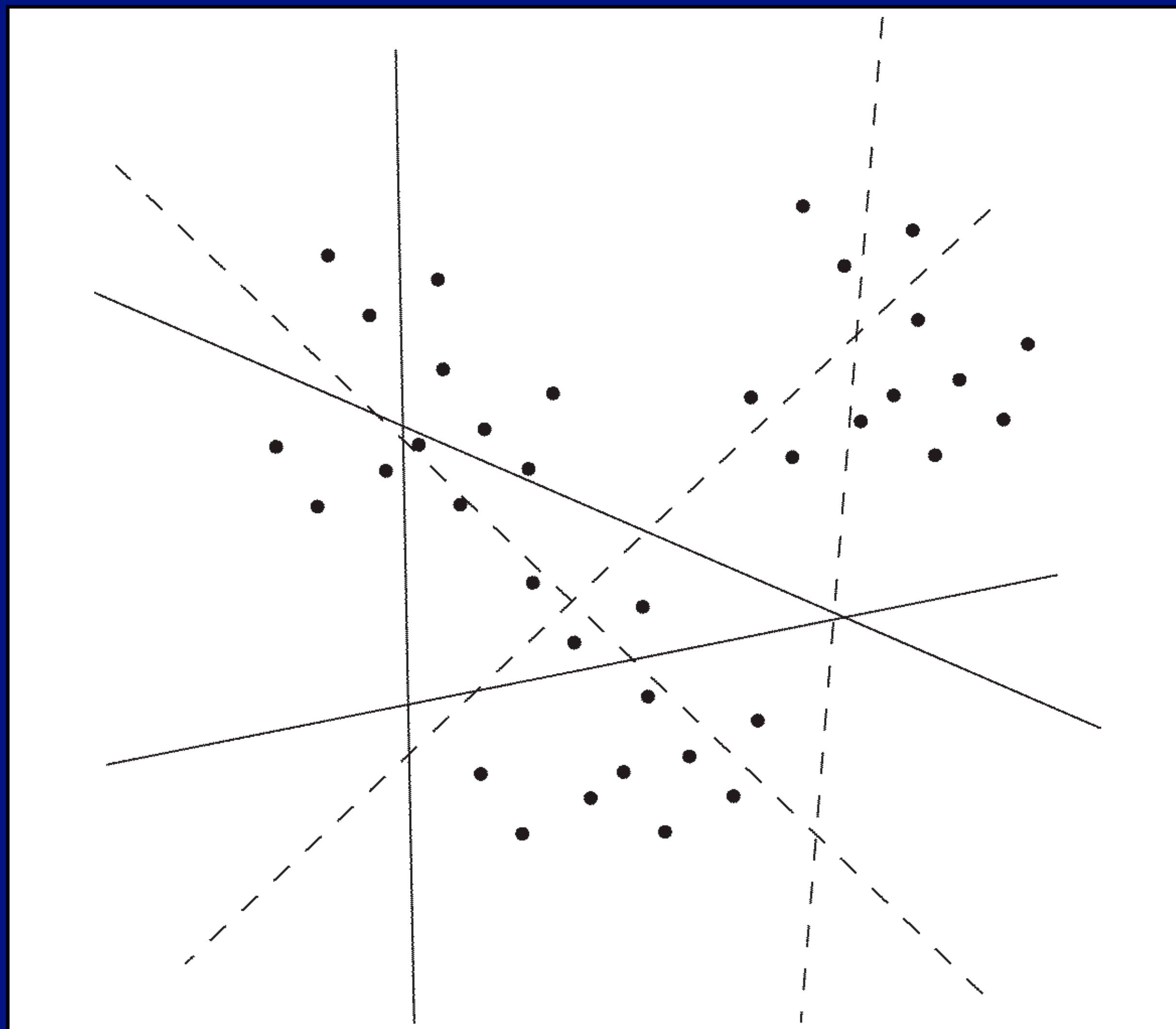
Locality sensitive hashing (LSH)

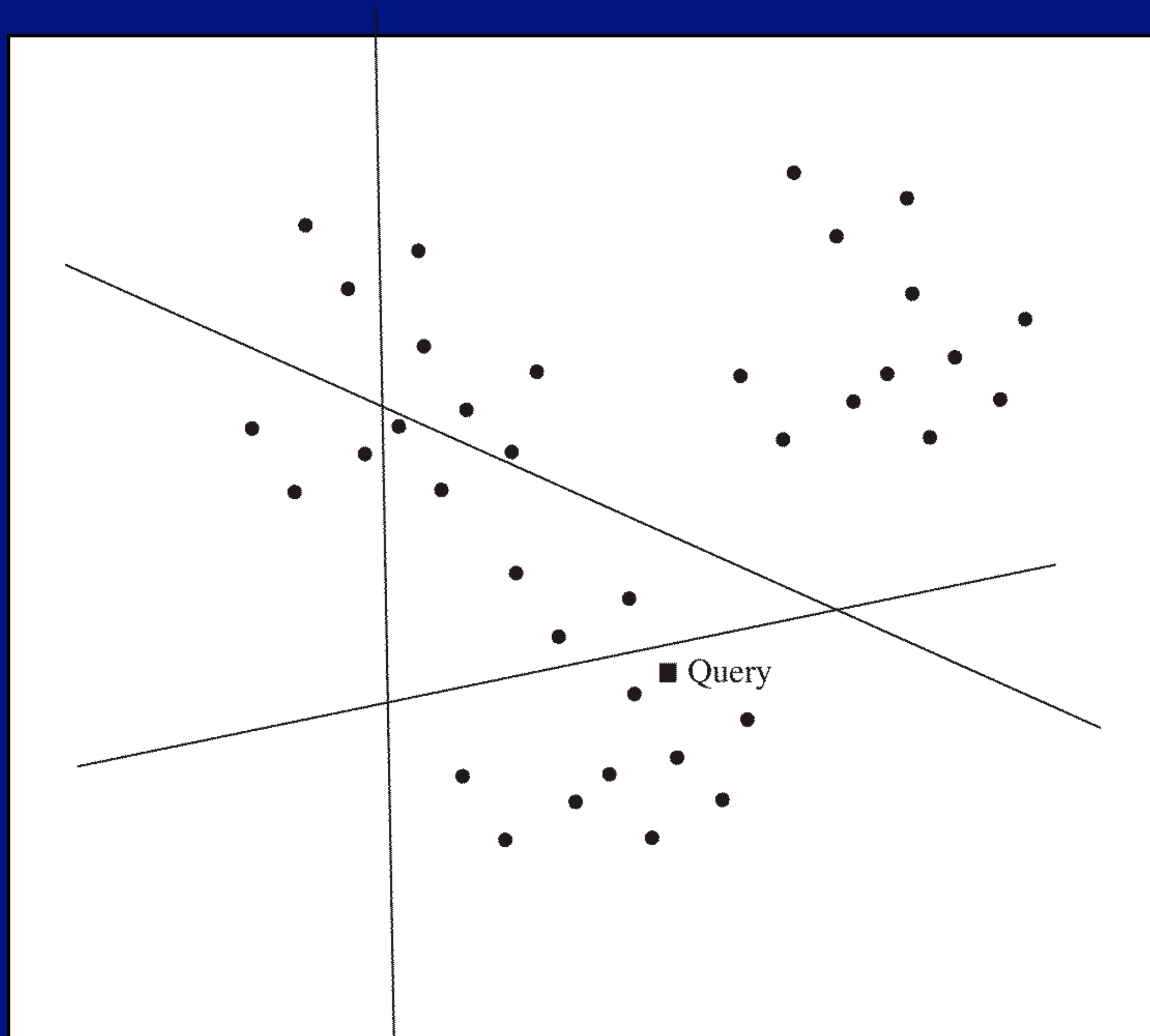
- Build a set of hash tables
- Insert each training data at its hash key
- ANN
  - compute key for test point
  - recover all points in each hash table at that key
  - linear search for distance in these points
  - take the nearest

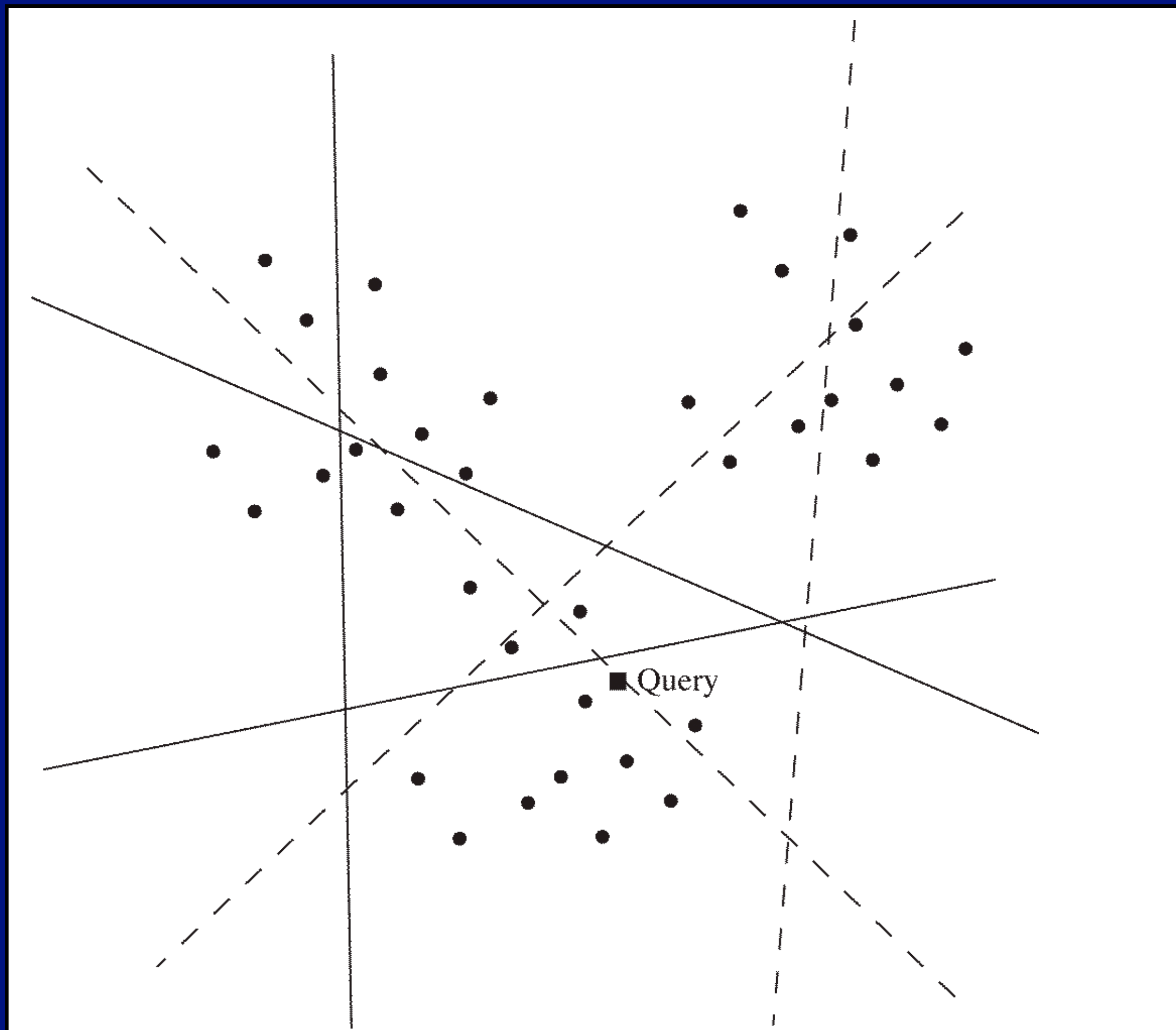
Hash functions

- Random splits
  - for each bit in the key, choose random w, b
  - bit is: sign (w*x+b)


2
+
101
111
1
+
001
011
+
3
110




Query


Query

LSH - issues

- Parameters
  - How many hash tables?
  - How many bits in key?
- Issues
  - quite good when data is spread out
  - can be weak when it is clumpy
    - too many points in some buckets, too few in others

kd-trees (outline)

- Build a kd-tree, splitting on median
- Walk the tree
  - find leaf in which query point lies
  - backtrack, pruning branches that are further away than best point so far

kd-Trees

- Standard construction fails in high dimensions
  - too much backtracking
- Good approximate nearest neighbor, if we
  - probe only a fixed number of leaves
  - use best bin first heuristic
- Very good for clumpy data

Approximate nearest neighbors

- In practice
  - fastest method depends on dataset
  - parameters depend on dataset
  - search methods, parameters using dataset
  - FLANN (http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN)
    - can do this search

Basic ideas in classifiers

- Loss
  - errors have a cost, and different types of error have different costs
    - this means each classifier has an associated risk
  - Total risk

$$R(s) = Pr\left\{1 \to 2 | \text{using } s\right\} L(1 \to 2) + Pr\left\{2 \to 1 | \text{using } s\right\} L(2 \to 1)$$

- Expected loss of classifying a point gives

1 if $p(1|\boldsymbol{x})L(1 \to 2) > p(2|\boldsymbol{x})L(2 \to 1)$

2 if $p(1|\boldsymbol{x})L(1 \to 2) < p(2|\boldsymbol{x})L(2 \to 1)$

Histogram based classifiers

- Represent class-conditional densities with histogram
- Advantage:
  - estimates become quite good
    - (with enough data!)
- Disadvantage:
  - Histogram becomes big with high dimension
    - but maybe we can assume feature independence?

Finding skin

- Skin has a very small range of (intensity independent) colours, and little texture
  - Compute an intensity-independent colour measure, check if colour is in this range, check if there is little texture (median filter)
  - See this as a classifier - we can set up the tests by hand, or learn them.

Histogram classifier for skin

$$\frac{P(rgb \mid skin)}{P(rgb \mid \neg skin)} \geq \Theta$$

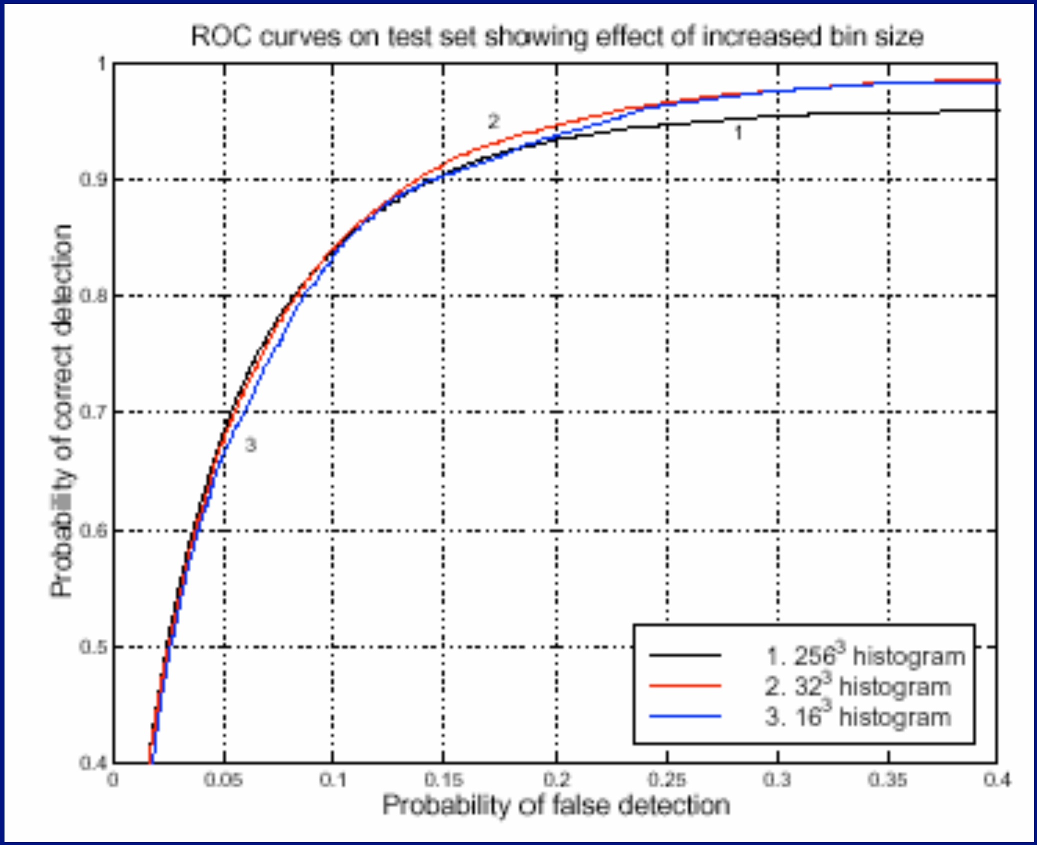


Figure from Jones+Rehg, 2002

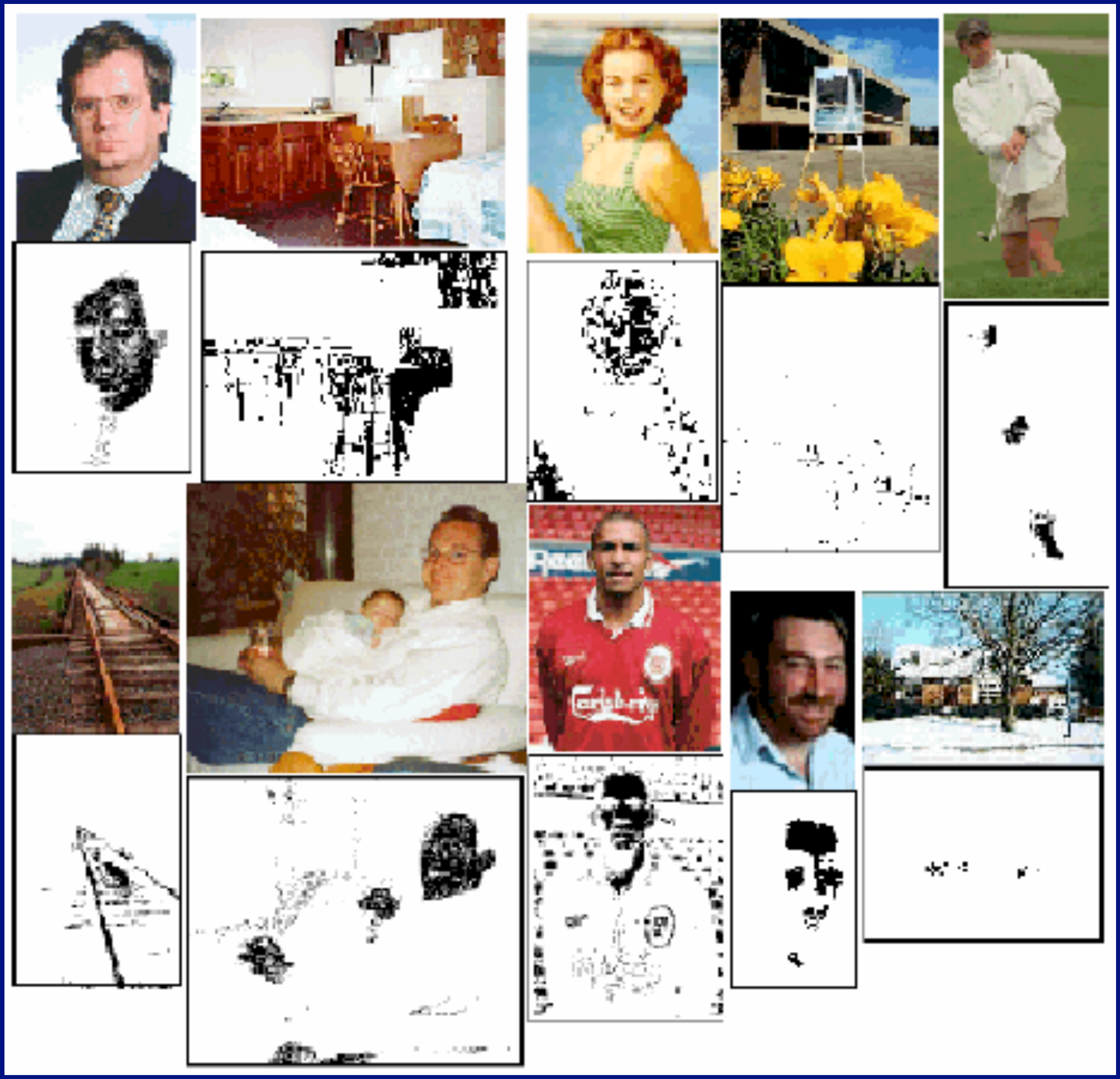


Figure from Jones+Rehg, 2002

Curse of dimension - I

- This won’t work for many features
  - try R, G, B, and some texture features
  - too many histogram buckets

Naive Bayes

- Previously, we detected with a likelihood ratio test

$$\frac{P(\text{features}|\text{event})}{P(\text{features}|\text{not event})} > \text{threshold}$$

- Now assume that features are conditionally independent given event

$$P(f_0, f_1, f_2, \ldots, f_n|\text{event}) = P(f_0|\text{event})P(f_1|\text{event})P(f_2|\text{event})\ldots P(f_n|\text{event})$$

Naive Bayes

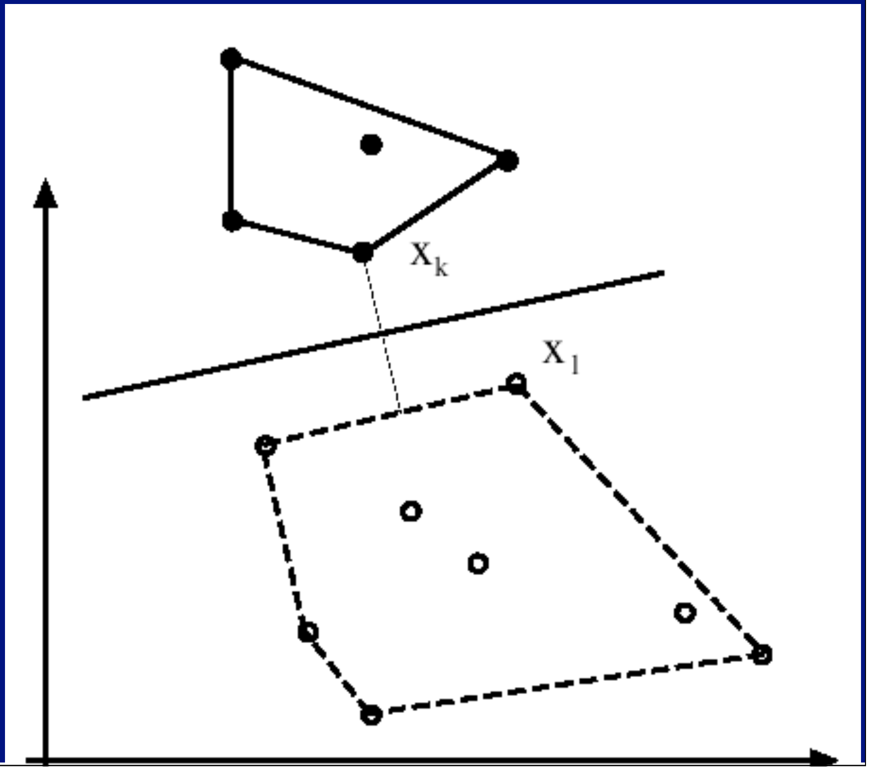
- (not necessarily perjorative)
- Histogram doesn’t work when there are too many features
  - the curse of dimension, first version
  - assume they’re independent conditioned on the class, cross fingers
  - reduction in degrees of freedom
  - very effective for face finders
    - relations may not be all that important
  - very effective for high dimensional problems
    - bias vs. variance

Logistic Regression

- Build a parametric model of the posterior,
  - p(class|information)
- For a 2-class problem, assume that
  - log(P(1|data))-log(P(0|data))=linear expression in data
- Training
  - maximum likelihood on examples
  - problem is convex
- Classifier boundary
  - linear

Decision boundaries

- The boundary matters
  - but the details of the probability model may not
- Seek a boundary directly
  - when we do so, many or most examples are irrelevant
- Support vector machine



Support Vector Machines, easy case

- Classify with sign(w.x+b)
- Linearly separable data means $y_i\,(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) > 0$
- Choice of hyperplane means $y_i\,(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1$
- Hence distance

$$\begin{aligned} dist(\boldsymbol{x}_k, \text{hyperplane}) + dist(\boldsymbol{x}_l, \text{hyperplane}) &= (\frac{\boldsymbol{w}}{|\boldsymbol{w}|}\cdot\boldsymbol{x}_k + \frac{b}{|\boldsymbol{w}|}) - (\frac{\boldsymbol{w}}{|\boldsymbol{w}|}\cdot\boldsymbol{x}_1 + \frac{b}{|\boldsymbol{w}|}) \\ &= \frac{\boldsymbol{w}}{|\boldsymbol{w}|}\cdot(\boldsymbol{x}_1 - \boldsymbol{x}_2) = \frac{2}{|\boldsymbol{w}|} \end{aligned}$$

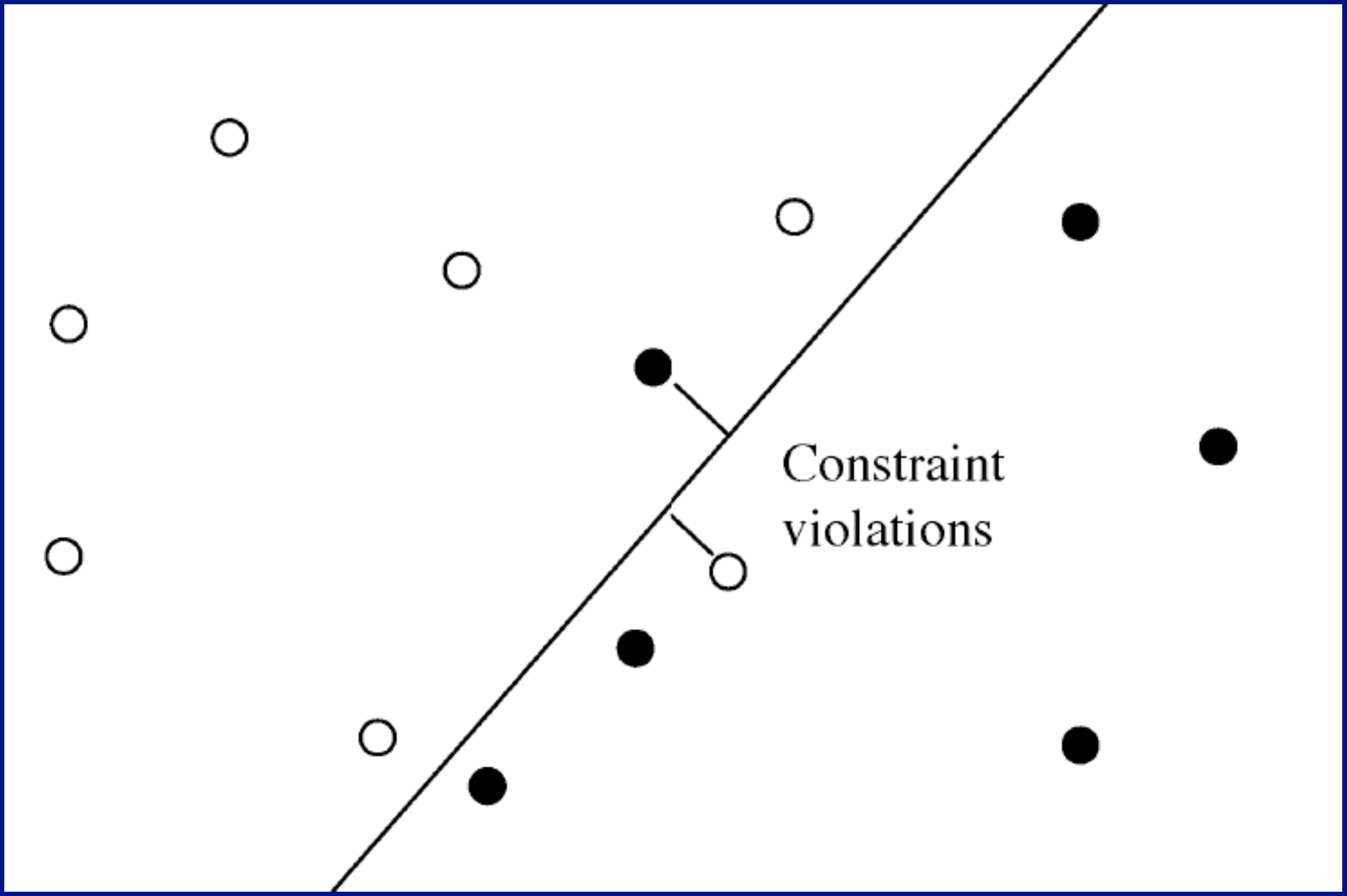
Support Vector Machines, separable case

$$\text{minimize} \quad (1/2)\boldsymbol{w} \cdot \boldsymbol{w}$$

$$\text{subject to} \quad y_i\left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right) \geq 1$$

By being clever about what x means, I can have much more interesting boundaries.

Data not linearly separable



Data not linearly separable

Objective function becomes

$$||\mathbf{w}||^2/2 + C\left(\sum_i \xi_i\right)$$

Constraints become

$$\begin{aligned}
\mathbf{x}_i \cdot \mathbf{w} + b &\geq +1 - \xi_i \quad \text{for } y_i = +1 \\
\mathbf{x}_i \cdot \mathbf{w} + b &\leq -1 + \xi_i \quad \text{for } y_i = -1 \\
\xi_i &\geq 0 \; \forall i.
\end{aligned}$$

SVM’s

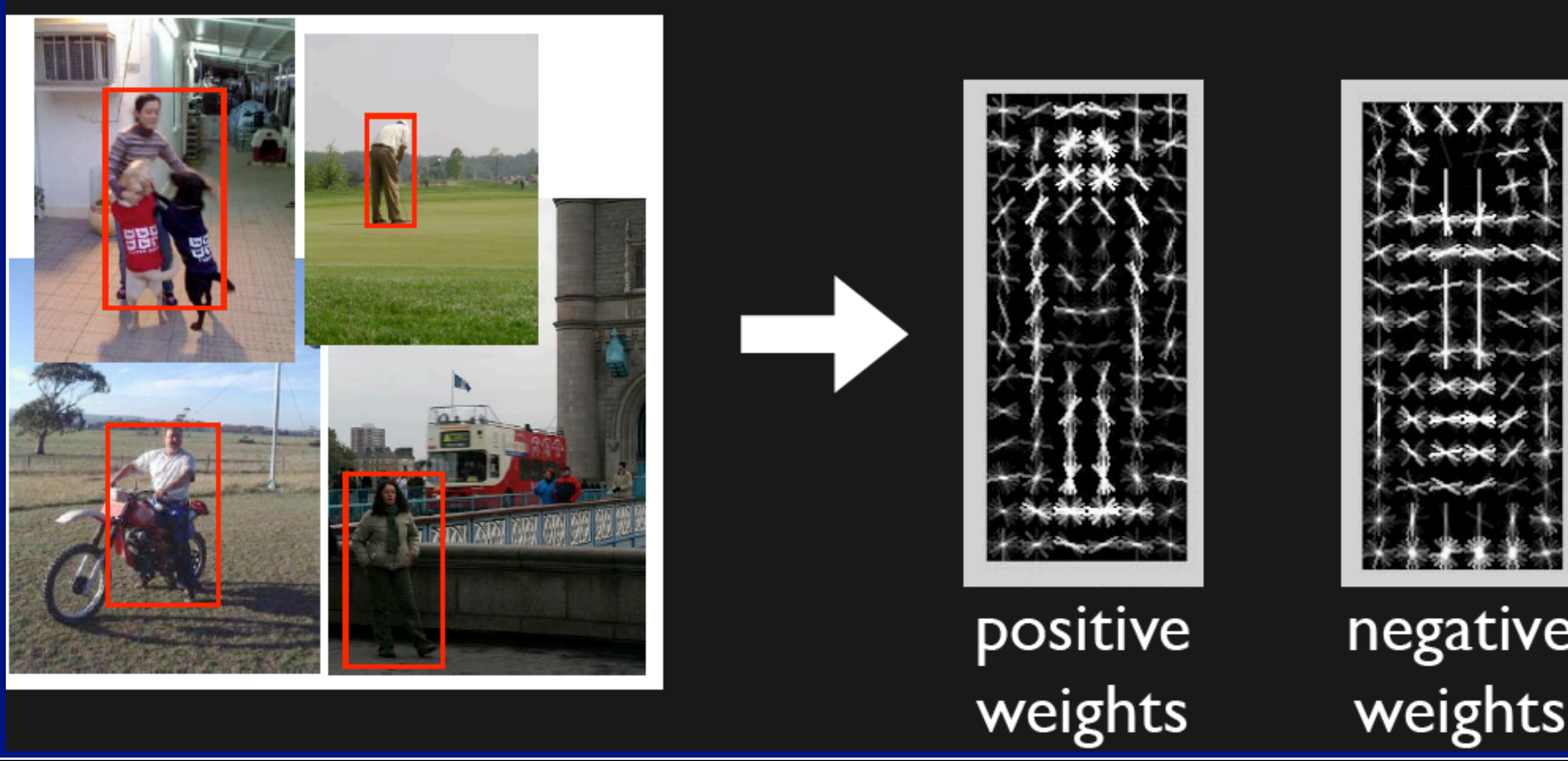
- Optimization problem is rather special
  - never ever use general purpose software for this
  - very well studied
- Methods available on the web
  - SVMlite
  - LibSVM
  - Pegasos
  - many others
- There are automatic feature constructions as well

Pedestrian detection with SVM

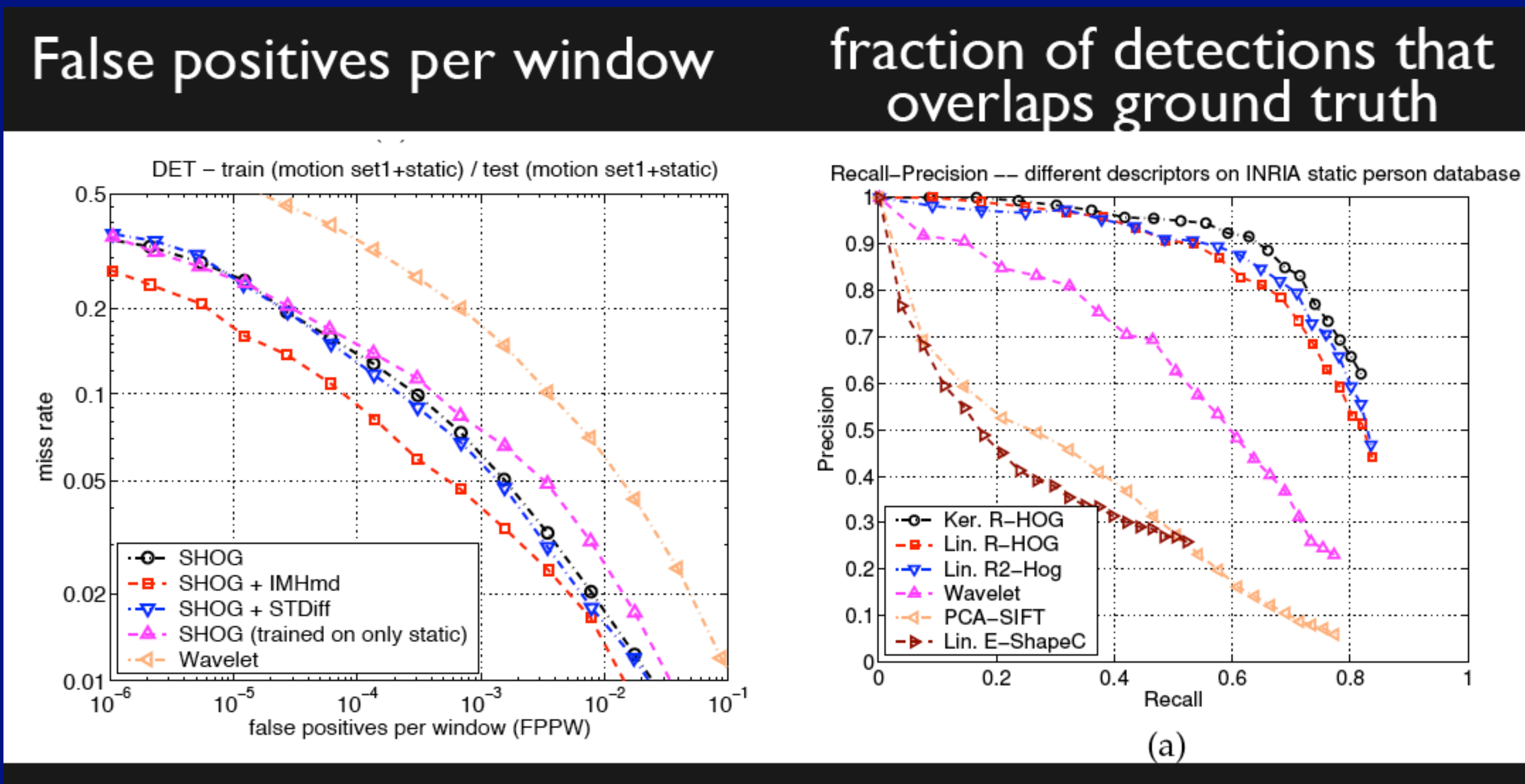
- Features:
  - HOG features in window
- Classifier:
  - Linear SVM
- Training:
  - pedestrian examples from INRIA
  - negative examples all over the place

SVM pedestrian detector

$$f_w(x) = w \cdot \Phi(x)$$



Pedestrian detectors: performance



Multiclass classification

- Many strategies
  - Easy with k-nearest neighbors
  - 1-vs-all
    - for each class, construct a two class classifier comparing it to all other classes
    - take the class with best output
      - if output is greater than some value
  - Multiclass logistic regression
    - log(P(i|features))-log(P(k|features))=(linear expression)
    - many more parameters
    - harder to train with maximum likelihood
    - still convex

Detection with a classifier

- Search
  - all windows
  - at relevant scales
- Prepare features
- Classify

- Issues
  - how to get only one response
  - speed
  - accuracy

Non-maximum suppression

- Compute "strength of response"
  - SVM value
  - LR value
- threshold
  - small values are not faces
- find largest value (over location, scale)
  - suppress nearby values
  - repeat

Speed+Accuracy

- There are methods to reject some windows early
  - using simple tests
- Key to good performance seems to be
  - very large numbers of examples
    - and clever methods to train with huge numbers of negatives
  - careful feature engineering

Near State of Art

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Our rank	3	1	2	1	1	2	2	4	1	1	1	4	2	2	1	1	2	1	4	1
Our score	.180	**.411**	.092	**.098**	**.249**	.349	.396	.110	**.155**	**.165**	**.110**	.062	.301	.337	**.267**	**.140**	.141	**.156**	.206	**.336**
Darmstadt							.301													
INRIA Normal	.092	.246	.012	.002	.068	.197	.265	.018	.097	.039	.017	.016	.225	.153	.121	.093	.002	.102	.157	.242
INRIA Plus	.136	.287	.041	.025	.077	.279	.294	.132	.106	.127	.067	.071	**.335**	.249	.092	.072	.011	.092	.242	.275
IRISA		.281					.318	.026	.097	.119			.289	.227	.221		**.175**			.253
MPI Center	.060	.110	.028	.031	.000	.164	.172	.208	.002	.044	.049	.141	.198	.170	.091	.004	.091	.034	.237	.051
MPI ESSOL	.152	.157	**.098**	.016	.001	.186	.120	**.240**	.007	.061	.098	**.162**	.034	.208	.117	.002	.046	.147	.110	.054
Oxford	**.262**	.409				**.393**	**.432**							**.375**					**.334**	
TKK	.186	.078	.043	.072	.002	.116	.184	.050	.028	.100	.086	.126	.186	.135	.061	.019	.036	.058	.067	.090

- Regular annual competition for best performing detectors
  - on a variety of categories
  - multiple competitors
- Typical results above from Deva Ramanan’s slides ’08

Crucial points

- Can detect some objects by
  - sliding window across image
  - computing features
  - presenting to classifier
- Works best for objects that are fairly rigid
  - pedestrians, faces, cars
- Classifier can be probabilistic or not
  - SVM is always first to try