Advanced Topics in Computer Vision
and Robotics

Classification Methods

Some slides thanks to S. Lazebnik, T. Berg, Fei-Fei Li, K. Grauman and others

Previously

- Object Recognition – Statistical Viewpoint

- Object Instance Recognition – Instance based methods

1. Local and global features
2. Quantization, visual vocabularies
3. Approximate nearest neighbour methods (k-d trees, k-means, hierarchical k-means of descriptors
4. Spatial Verification

Locally Sensitive Hashing

- Another methods for Approximate Nearest Neighbour
- Application: Large Scale Image Retrieval
- Even faster look up then binary trees, randomized algorithm

- Find $\mathbf{x}_j$ which is within the radius r of the query point, with a high probability
- Need a hash function $g(x)$ such that two points which are near by will hash to the same code
- Assume that each point is a bit string 0 1 0 1 0 0 ….
- Intuition – bin the axes, and the points which fall in the same bin are likely to be close, how about higher dimension ?

Locally Sensitive Hashing

- Idea: create several random projections and combine the results
- Create l hash tables $g_1(x), g_2(x) \cdots g_l(x)$
- Each will model one projection of the point
- Enter all the examples in the hash tables
- For query point, get all the examples which are in the same bin for all hash function, take union of these points
- Those will be the candidates

- Application; database of 13 million web images, image descriptor 512 dimensions, LSH examines only thousands examples (thousand fold speedup over exhaustive or k-d tree)

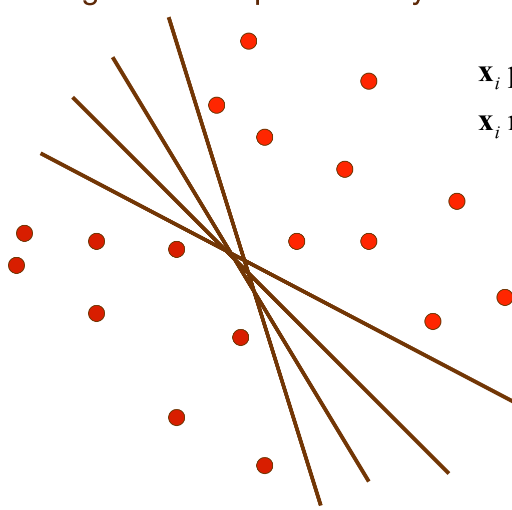
Ensemble Methods

Support Vector Machines,

Slides from S. Lazebnik, adopted slides from A. Moore

Linear classifiers

- Find linear function (*hyperplane*) to separate positive and negative examples – many such hyperplanes



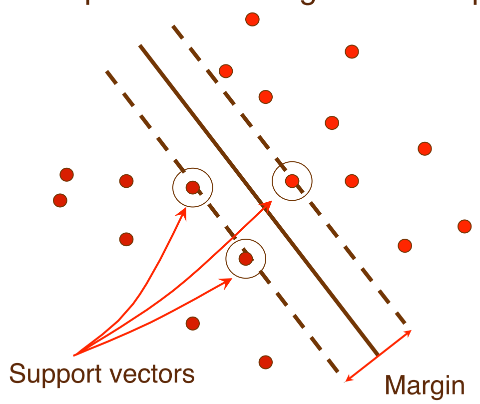
$\mathbf{x}_i$ positive: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

$\mathbf{x}_i$ negative: $\quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$

Which hyperplane is best?

Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support, vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

1. Maximize margin $2/\|\mathbf{w}\|$
2. Correctly classify all training data:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$
$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

- *Quadratic optimization problem*:

Minimize $\quad \dfrac{1}{2}\mathbf{w}^T\mathbf{w}$

Subject to $\; y_i(\mathbf{w} \cdot \boldsymbol{x}_i + b) \geq 1$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Solving the Optimization Problem

Quadratic programming with linear constraints

$$\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge 1$$

Lagrangian Function

$$\text{minimize } L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t. } \alpha_i \ge 0$$

Slide credit: Jinwei G

Solving the Optimization Problem

$$\text{minimize } L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t. } \alpha_i \ge 0$$

Lagrangian Dual Problem

$$\text{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

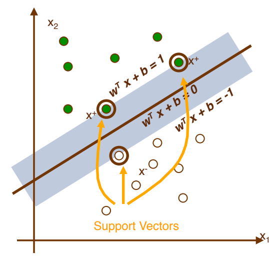
$$\text{s.t. } \alpha_i \ge 0 \text{, and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

Slide credit: Jinwei G

Solving the Optimization Problem

- From KKT condition, we know:

$$\alpha_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 \right) = 0$$



- Thus, only support vectors have $\alpha_i \neq 0$

- The solution has the form:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$

get b from $y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0,$

where $\mathbf{x}_i$ is support vector

Slide credit: Jinwei G

Finding the maximum margin hyperplane

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

learned weight

Support vector

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

  $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ for any support vector

- Classification function (decision boundary):

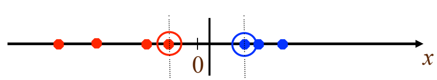
$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x}_i$
- Solving the optimization problem also involves computing the inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ between all pairs of training points

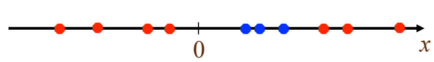
C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Nonlinear SVMs

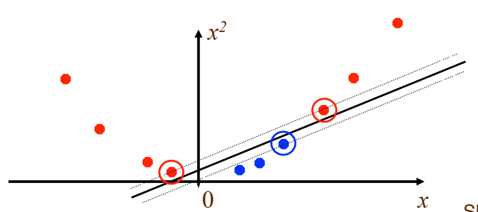
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



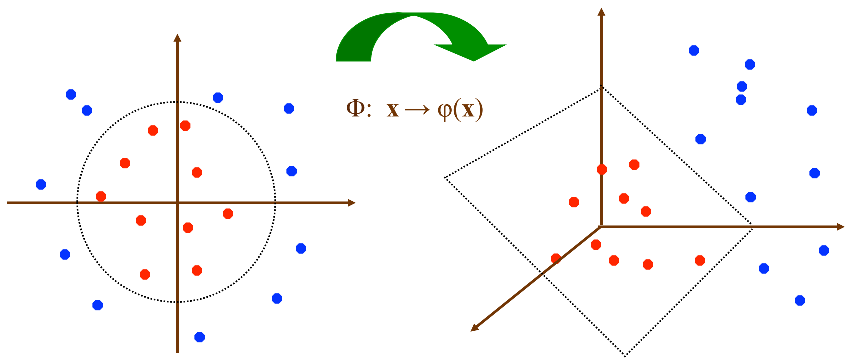
- We can map it to a higher-dimensional space:



Slide credit: Andrew Moore

Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Slide credit: Andrew Moore

Recall Solving the Optimization Problem

- The linear discriminant function is:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i \in SV} \alpha_i \mathbf{x}_i^T \mathbf{x} + b$$

- Notice it relies on a *dot product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x_i}$

Slide credit: Jinwei G

Nonlinear SVM: Optimization

- Formulation: (Lagrangian Dual Problem)

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

such that

$$0 \le \alpha_i \le C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

- The solution of the discriminant function is

$$g(\mathbf{x}) = \sum_{i \in SV} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$

- The optimization technique is the same.

Slide credit: Jinwei G

Nonlinear SVMs – Kernel Trick

- With this mapping, our discriminant function is now:

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in SV} \alpha_i \boxed{\phi(\mathbf{x}_i)^T \phi(\mathbf{x})} + b$$

- No need to know this mapping explicitly, because we only use the dot product of feature vectors in both the training and test.

- A *kernel function* is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Slide credit: Jinwei G

Nonlinear SVMs – Kernel Trick

- Examples of commonly-used kernel functions:
  - Linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
  - Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
  - Gaussian (Radial-Basis Function (RBF)) kernel:

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

  - Sigmoid:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$$

Slide credit: Jinwei G

Kernels for bags of features

- Histogram intersection kernel:

$$I(h_1, h_2) = \sum_{i=1}^{N} \min(h_1(i), h_2(i))$$

- Generalized Gaussian kernel:

- $$K(h_1, h_2) = \exp\left(-\frac{1}{A} D(h_1, h_2)^2\right)$$

- *D* can be Euclidean distance, χ^2 distance, Earth Mover's Distance, etc.

J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, Local Features and Kernels for Classifcation of Texture and Object Categories: A Comprehensive Study, IJCV 2007

Support Vector Machine: Algorithm

- 1. Choose a kernel function
- 2. Choose a value for *C – bound on maximum weight for each support vector*
- 3. Solve the quadratic programming problem (many software packages available)
- 4. Construct the discriminant function from the support vectors

Slide credit: Jinwei G

Some Issues

- Choice of kernel
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed
  - domain experts can give assistance in formulating appropriate similarity measures
- Choice of kernel parameters
  - e.g. σ in Gaussian kernel
  - σ is the distance between closest points with different classifications
  - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.

This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Slide credit: Jinwei G

Summary: Support Vector Machine

- 1. Large Margin Classifier
  - Better generalization ability & less over-fitting
- 2. The Kernel Trick
  - Map data points to higher dimensional space in order to make them linearly separable.
  - Since only dot product is used, we do not need to represent the mapping explicitly.

Slide credit: Jinwei G

Summary: SVMs for image classification

1. Pick an image representation (in our case, bag of features)
2. Pick a kernel function for that representation
3. Compute the matrix of kernel values between every pair of training examples
4. Feed the kernel matrix into your favorite SVM solver to obtain support vectors and weights
5. At test time: compute kernel values for your test example and each support vector, and combine them with the learned weights to get the value of the decision function

What about multi-class SVMs?

- Unfortunately, there is no “definitive” multi-class SVM formulation
- In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs
- One vs. others
  - Traning: learn an SVM for each class vs. the others
  - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- One vs. one
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM “votes” for a class to assign to the test example

SVMs: Pros and cons

- Pros
  - Many publicly available SVM packages: http://www.kernel-machines.org/software
  - Kernel-based framework is very powerful, flexible
  - SVMs work very well in practice, even with very small training sample sizes
- Cons
  - No “direct” multi-class SVM, must combine two-class SVMs
  - Computation, memory
    - During training time, must compute matrix of kernel values for every pair of examples
    - Learning can take a very long time for large-scale problems

Multi-class classification

- How to deal with multiple classes
- One vs. all strategy
- For N classes train N different classifiers
- For class 1 positive examples – others negative examples
- How to combine the classifiers ?
- Each will output some confidence score $h_\theta(x)$
- Final prediction will be the class with highest confidence score

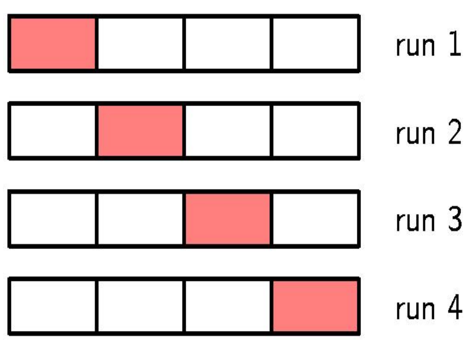
Bias and Variance

- Modeling issues: overfiting, underfiting
- How do you know how good is your model
- Example: regression (linear, vs 3rd order polynomial)
- Intuition: models which underfit have large bias
- Models which overfit have large variance
- Idea: fit the model to different subsets of data, if we fit the line that line will have roughly similar parameters, but large test error – small variance, large bias
- If we fit overfit, each model will have small error but the parameters of the model will have large variance

Bias and Variance

- Modeling issues: overfiting, underfiting in classification
- How do you know how good is your model
- 0/1 classification error: proportion of misclassified examples
- Training error and test error
- Picture of variance bias trade-off: curvature of decision boundary
- How do you choose good model in practice ?
- Hold-out-cross-validation
- Split data into 70% train and 30% cross-validation
- Generate N different models, pick the one with lowest error on cross-validation set

Cross-validation: Example



4-fold cross-validation

It allows to use ¾ of the available data for training, while making use of all of the data to assess performance

k-fold Cross-validation

- In general: we perform k runs. Each run uses (k-1)/k of the available data for training.
- If the number of data is very limited, we can set k=N (total number of data points). This gives the ***leave-one-out*** cross-validation technique.

k-fold Cross-validation: Drawbacks

- Computationally expensive: number of training runs is increased by a factor of k.
- A single models may have multiple complexity parameters: exploring combinations of settings could require a number of training runs that is *exponential* in the number of parameters.

In practice

- What are the choices if the first choice does not work ?
- i.e. large generalization error
- If the model has high bias – it is too simple: consider adding more features or using deeper decision tree
- If the model has high variance – it is too complex: fits the idiosyncracy of the data: remove features, or get more data

Ensemble Methods

- So far we considered only single hypothesis
- Methods which consider whole ensemble of hypotheses , from some hypothesis space and combine their prediction
- One idea – consider majority vote N=5 hypotheses, if at least 3 classify it correctly then it is correct label (majority vote)
- Example: multiple separating lines for non-linearly separable classes, note individual hypotheses are simple
- Most popular idea: Boosting

Boosting

- Defines a classifier using an additive model:

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \dots$$

Strong classifier

Features vector

Weight

Weak classifier

- We need to define a family of weak classifiers

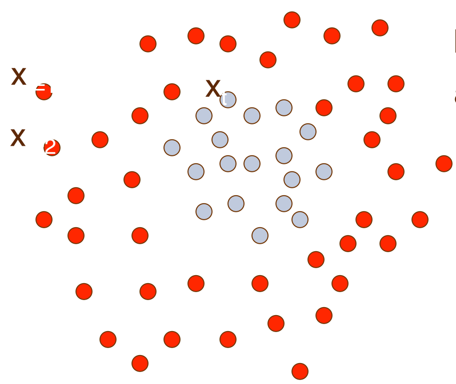
$f_k(x)$ from a family of weak classifiers

Boosting

- Each training example has associated w_i
- At the beginning all $w_i = 1$
- Generate first hypothesis h_1 , some example correct, some no
- Idea: next do better of misclassified examples
- Increase weights of misclassified examples, decrease weight of correctly classified examples, etc …
- Final ensemble is weighted majority combination of all examples

Boosting

- It is a sequential procedure:



Each data point has

a class label:

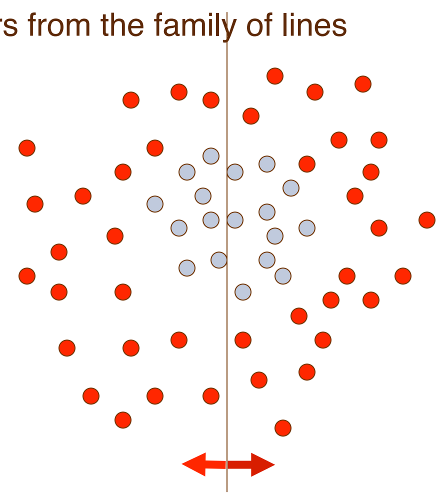
$$y = \begin{cases} +1 & (\ ) \\ -1 & (\ ) \end{cases}$$

and a weight:
w =1

Slide credit: Antonio Torralba

Toy example

Weak learners from the family of lines



Each data point has

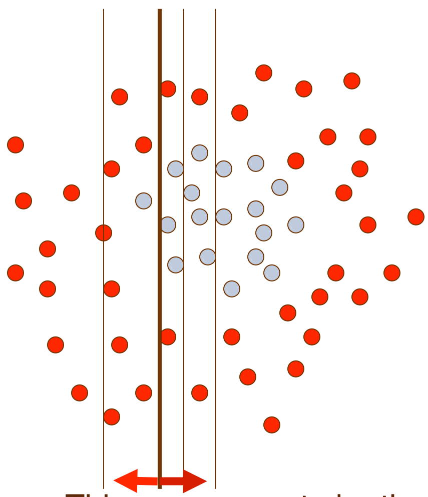
a class label:

and a weight:
w =1

h => p(error) = 0.5 it is at chance

Slide credit: Antonio Torralba

Toy example



Each data point has

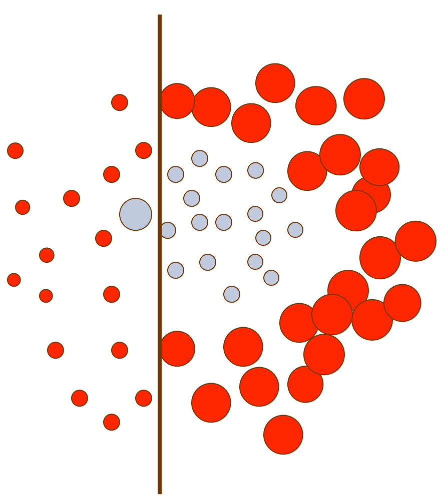
a class label:

and a weight:
w =1

This one seems to be the best
This is a ‘**weak classifier**’: It performs slightly better than chance.

Slide credit: Antonio Torralba

Toy example



Each data point has

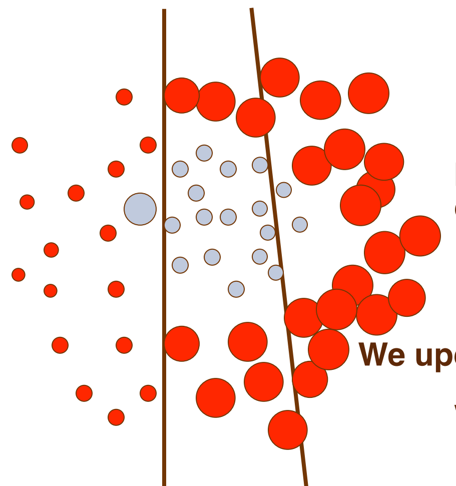
a class label:

We update the weig
$w \leftarrow w \exp\{-y H\}$

We set a new problem for which the previous weak classifier performs at chance again

Slide credit: Antonio Torralba

Toy example



Each data point has a class label:

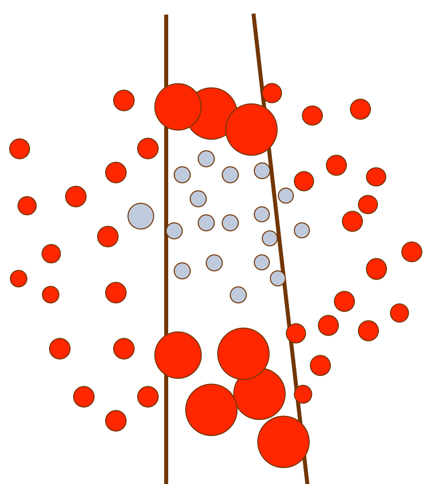
We update the weights:

$w \leftarrow w \exp\{-y H\}$

We set a new problem for which the previous weak classifier performs at chance again

Slide credit: Antonio Torralba

Toy example



Each data point has

a class label:

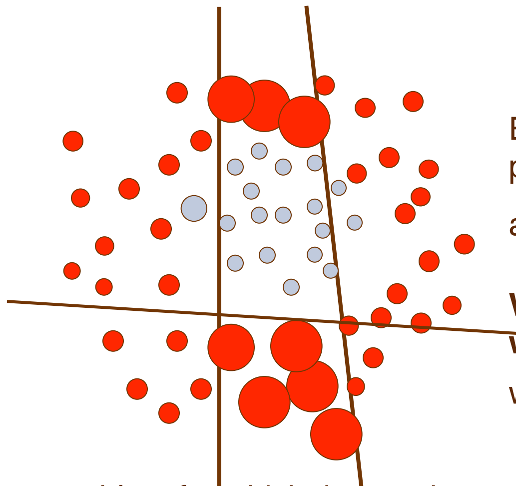
We update the weights:

$w \leftarrow w \exp\{-y H\}$

We set a new problem for which the previous weak classifier performs at chance again

Slide credit: Antonio Torralba

Toy example



Each data point has

a class label:

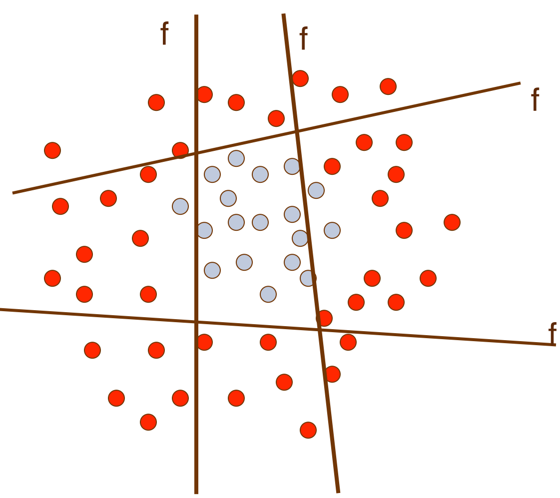
We update the weights:

w ← w exp{-y H}

We set a new problem for which the previous weak classifier performs at chance again

Slide credit: Antonio Torralba

Toy example



The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

Slide credit: Antonio Torralba

Adaboost

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$.
- Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

Slide credit: Antonio Torralba

Boosting

- Advantages of boosting
  - Integrates classification with feature selection
  - Complexity of training is linear instead of quadratic in the number of training examples
  - Flexibility in the choice of weak learners, boosting scheme
  - Testing is fast
  - Easy to implement
- Disadvantages
  - Needs many training examples
  - Often doesn't work as well as SVM (especially for many-class problems)

AdaBoost learning algorithm

Discrete AdaBoost(Freund & Schapire 1996*b*)

1. Start with weights $w_i = 1/N$, $i = 1, \dots, N$.
2. Repeat for $m = 1, 2, \dots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on the training data.
   (b) Compute $\mathrm{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m \cdot 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \dots N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\mathrm{sign}[\sum_{m=1}^{M} c_m f_m(x)]$

47

Generic category recognition: basic framework

- Build/train object model
  - Choose a representation
  - Learn or fit parameters of model / classifier
- Generate candidates in new image
- Score the candidates

Window-based models
Building an object model

Given the representation, train a binary classifier

Car/non-car Classifier

Yes, car.

No, not a car.

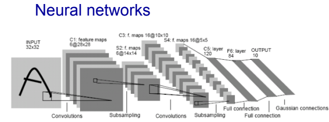
Discriminative classifier construction

Nearest neighbor

10^6 examples

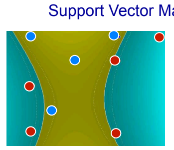
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



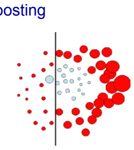
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Support Vector Machines



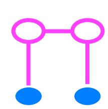
Guyon, Vapnik
Heisele, Serre, Poggio, 2001,…

Boosting



Viola, Jones 2001, Torralba et al. 2004, Opelt et al. 2006,…

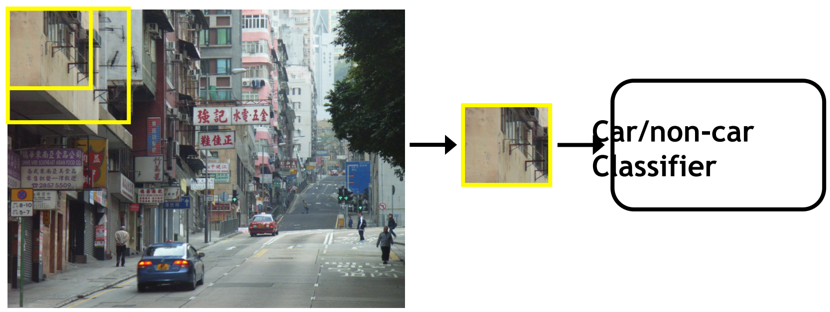
Conditional Random Fields



McCallum, Freitag, Pereira 2000; Kumar, Hebert 2003
…

Slide adapted from Antonio Torralba

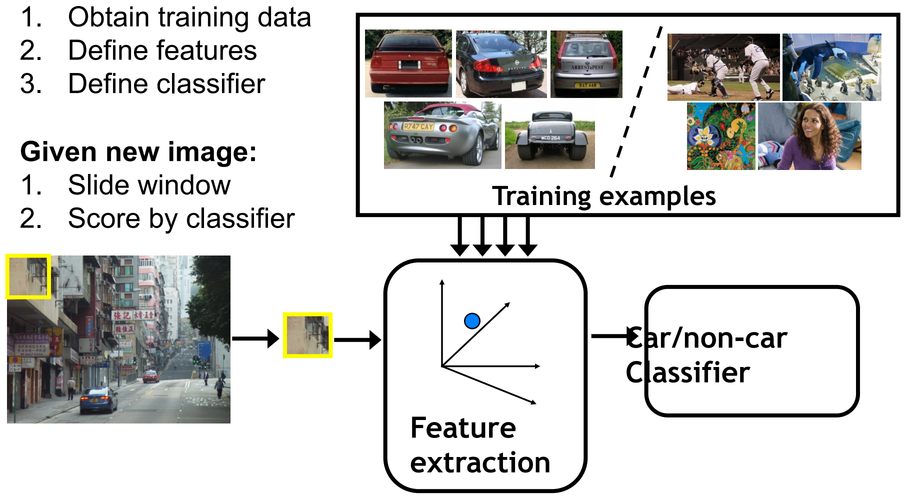
Window-based models
Generating and scoring candidates



Window-based object detection: recap

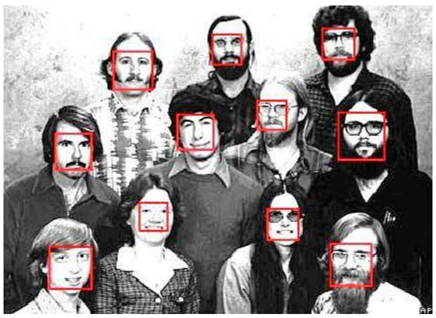
Training:
1. Obtain training data
2. Define features
3. Define classifier

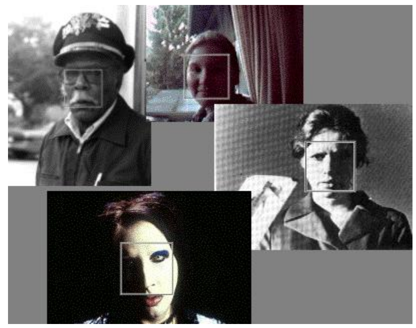
Given new image:
1. Slide window
2. Score by classifier



Kristen Grauman

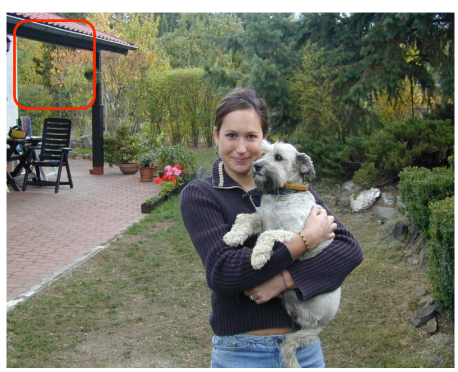
Face detection



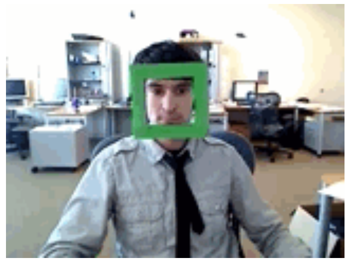


Face detection

- Basic idea: slide a window across image and evaluate a face model at every location

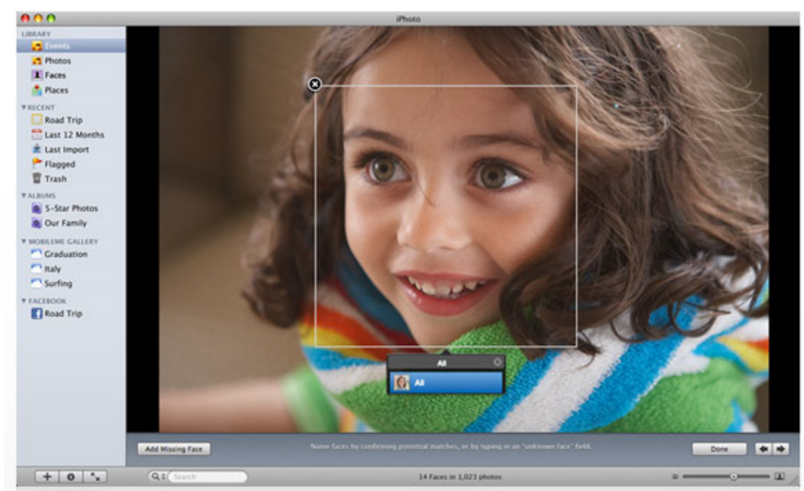


Face detection



Behold a state-of-the-art face detector!
(Courtesy Boris Babenko)

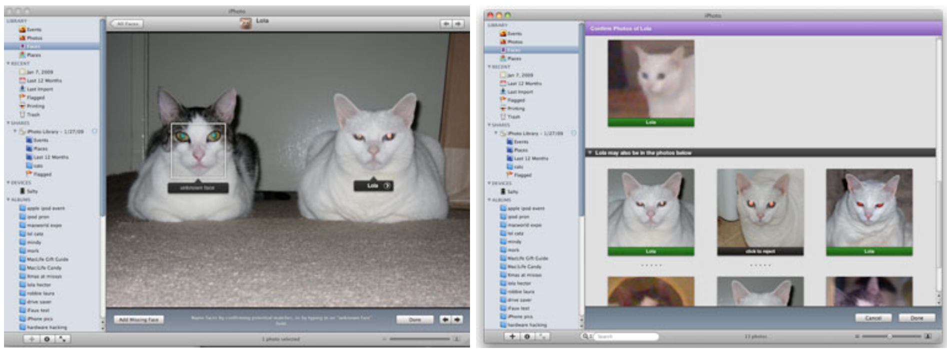
Consumer application: Apple iPhoto



http://www.apple.com/ilife/iphoto/

Consumer application: Apple iPhoto

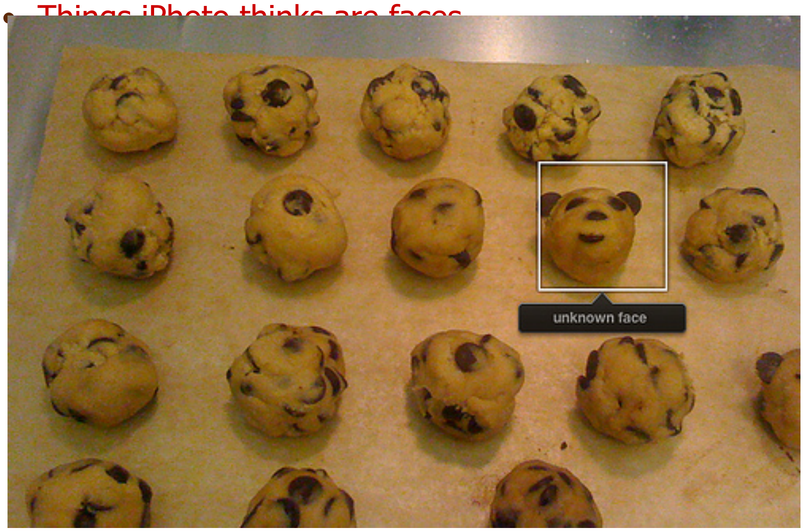
- Can be trained to recognize pets!



http://www.maclife.com/article/news/iphotos_faces_recognizes_cats

Consumer application: Apple iPhoto

- Things iPhoto thinks are faces

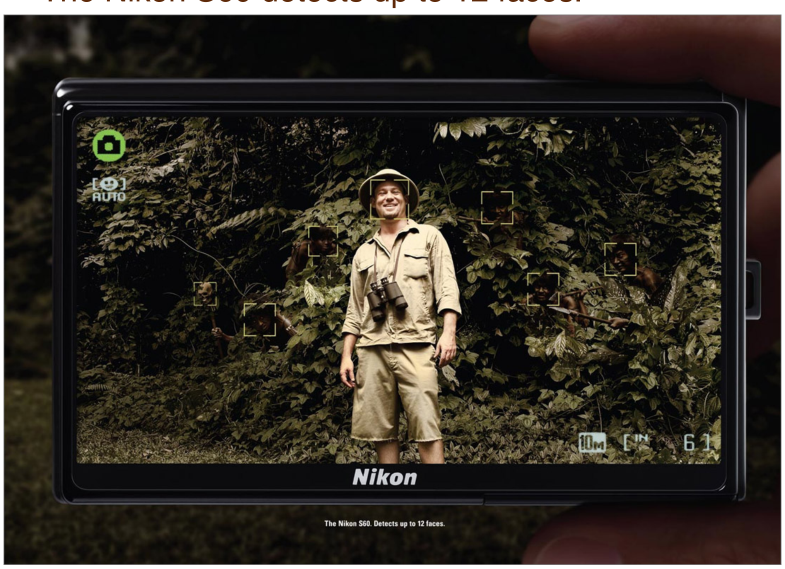


Funny Nikon ads

"The Nikon S60 detects up to 12 faces."

Funny Nikon ads

"The Nikon S60 detects up to 12 faces."



Challenges of face detection

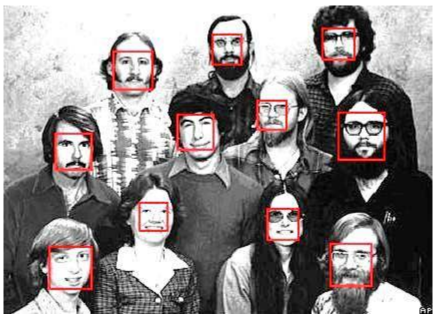
- Sliding window detector must evaluate tens of thousands of location/scale combinations
- Faces are rare: 0–10 per image
  - For computational efficiency, we should try to spend as little time as possible on the non-face windows
  - A megapixel image has ~10^6 pixels and a comparable number of candidate face locations
  - To avoid having a false positive in every image image, our false positive rate has to be less than 10^{-6}

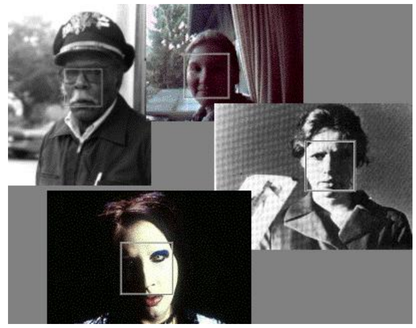
The Viola/Jones Face Detector

- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.
P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

Face detection





Challenges of face detection

- Sliding window detector must evaluate tens of thousands of location/scale combinations
- Faces are rare: 0–10 per image
  - For computational efficiency, we should try to spend as little time as possible on the non-face windows
  - A megapixel image has ~10^6 pixels and a comparable number of candidate face locations
  - To avoid having a false positive in every image image, our false positive rate has to be less than 10^{-6}

The Viola/Jones Face Detector

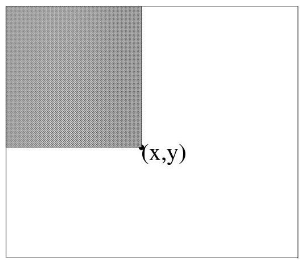
- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

A totally different idea

- Use many very simple features
- Learn cascade of tests for target object
- Efficient if:
  - features easy to compute
  - cascade short

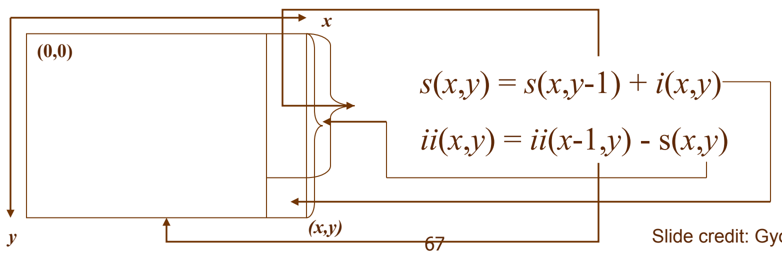
Integral Image



Def: The *integral image* at location (x,y), is the sum of the pixel values above and to the left of (x,y), inclusive. We can calculate the integral image representation of the image in a single pass.

ii(x,y) – value of the integral image – sum of all pixels above and left of (x,y)

s(x,y) – cummulative row sum



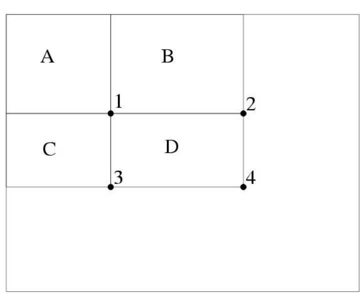
$s(x,y) = s(x,y\text{-}1) + i(x,y)$

$ii(x,y) = ii(x\text{-}1,y)$ - $s(x,y)$

67

Slide credit: Gyozo Gidofalvi

Efficient Computation of Rectangle Value



Using the integral image representation one can compute the value of any rectangular sum in constant time.

Example: Rectangle D

$ii(4) + ii(1) – ii(2) – ii(3)$

As a result two-, three-, and four-rectangular features can be computed with 6, 8 and 9 array references respectively.

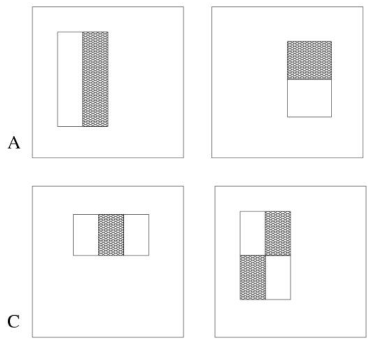
Idea: Compute lot of simple features – outputs of convolution with the box like filters

Object detection: classification problem

68

Using Many Simple Features

- Viola Jones / Haar Features

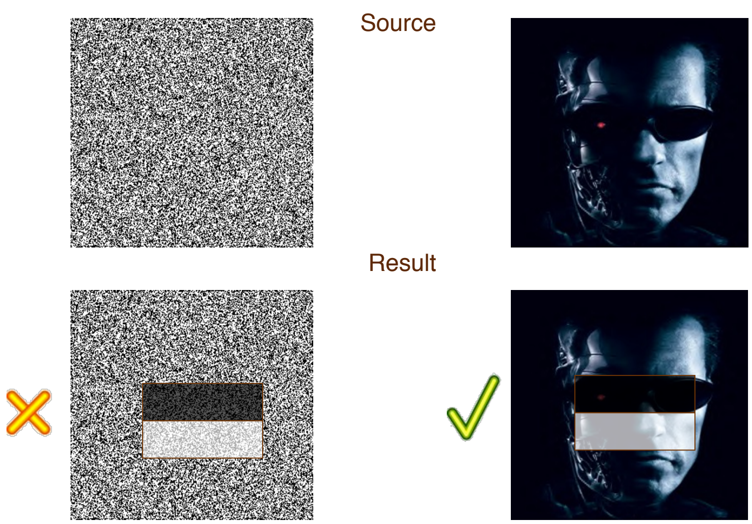


<u>(Generalized) Haar Features:</u>

- rectangular blocks, white or black
- 3 types of features:
  - two rectangles: horizontal/vertical
  - three rectangles
  - four rectangles
- in 24x24 window: 180,000 possible features

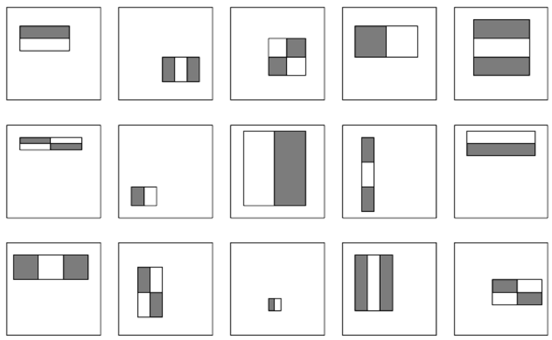
69

Example



Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!



Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!
- At test time, it is impractical to evaluate the entire feature set
- Can we create a good classifier using just a small subset of all possible features?
- How to select such a subset?

Boosting

- Boosting is a classification scheme that works by combining *weak learners* into a more accurate ensemble classifier
  - A weak learner need only do better than chance
- Training consists of multiple *boosting rounds*
  - During each boosting round, we select a weak learner that does well on examples that were hard for the previous weak learners
  - "Hardness" is captured by weights attached to training examples

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

Problem

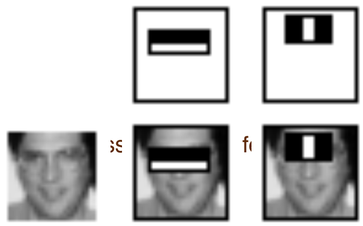
- For a 24x24 detection region, the number of possible rectangle features is ~160,000!
- At test time, it is impractical to evaluate the entire feature set
- Can we create a good classifier using just a small subset of all possible features?
- How to select such a subset?
- Feature here is equivalent with weak hypothesis

- Answer: Boosting [AdaBoost, Freund/Shapire]
  - Finds small set of features that are "sufficient"
  - Generalizes very well
  - Requires positive and negative examples

74

AdaBoost Idea (in Viola/Jones):

- Given set of “weak” classifiers:
  - Pick best one
  - Reweight training examples, so that misclassified images have larger weight
  - Reiterate; then linearly combine resulting classifiers



75

Boosting for face detection

- Define weak learners based on rectangle features

value of rectangle feature

$$h_t(x) = \begin{cases} 1 & \text{if } p_t f_t(x) > p_t \theta_t \\ 0 & \text{otherwise} \end{cases}$$

window

parity

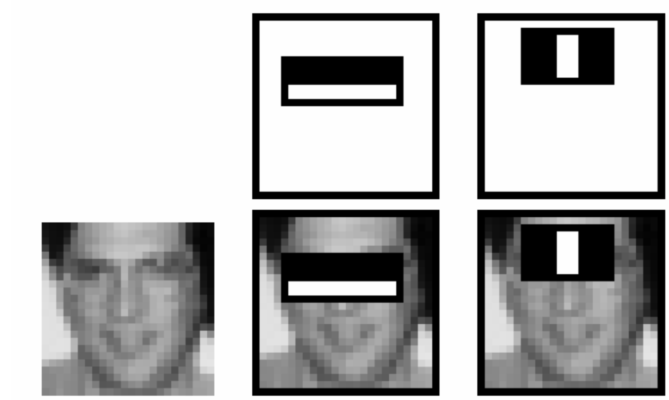
threshold

Boosting for face detection

- Define weak learners based on rectangle features
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Select best threshold for each filter
  - Select best filter/threshold combination
  - Reweight examples
- Computational complexity of learning: $O(MNK)$
  - M rounds, N examples, K features

Boosting for face detection

- First two features selected by boosting:
-

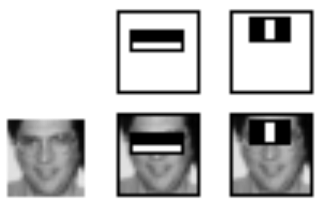


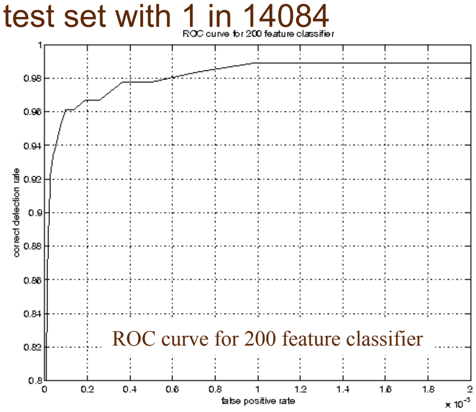
This feature combination can yield 100% detection rate and 50% false positive rate

Example Classifier for Face Detection

A classifier with 200 rectangle features was learned using AdaBoost

95% correct detection on test set with 1 in 14084 false positives.

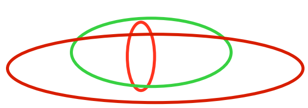


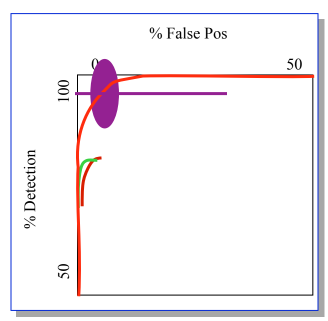


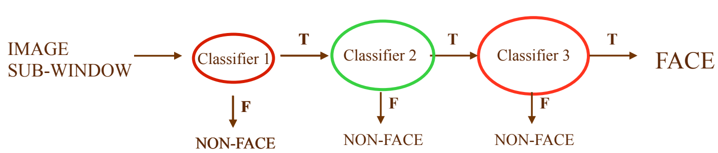
Slide credit: Frank Dellaert, Paul Viola, Foryth&Ponce

Classifier are Efficient

- Given a nested set of classifier hypothesis classes

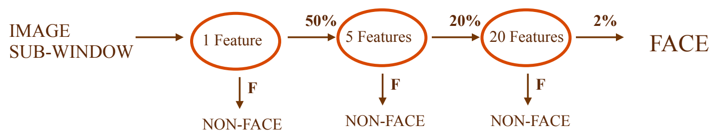






80Slide credit: Frank Dellaert, Paul Viola, Forsyth&Ponce

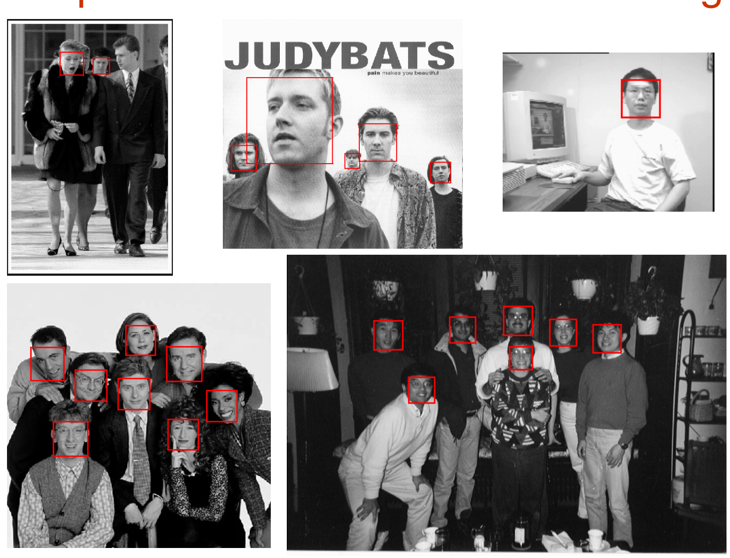
Cascaded Classifier



- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - using data from previous stage.
- A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)

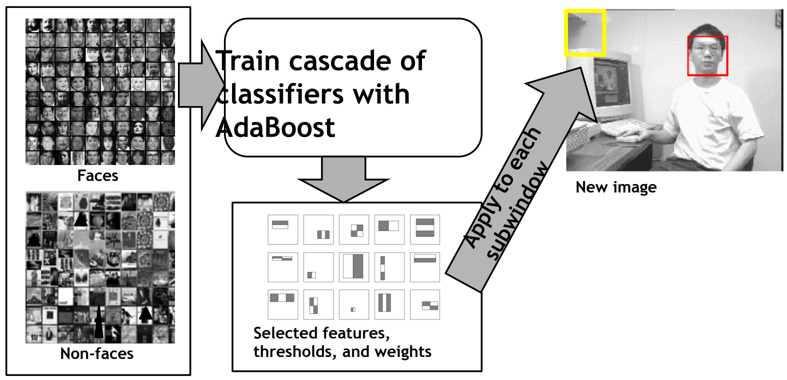
 Slide credit: Frank Dellaert, Paul Viola, Foryth&Ponce

Output of Face Detector on Test Images



 Slide credit: Frank Dellaert, Paul Viola, Foryth&Ponce

Viola-Jones detector: summary

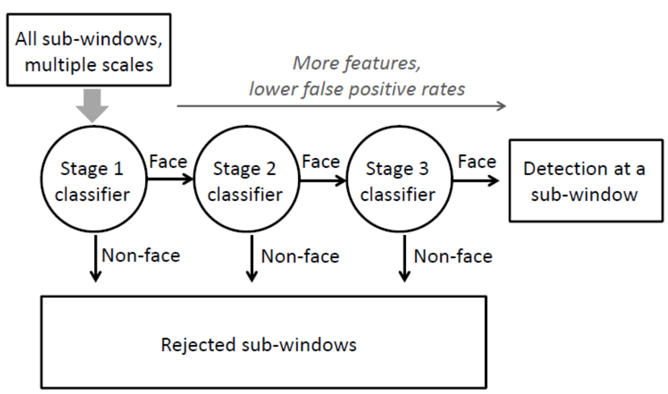


Train with 5K positives, 350M negatives
Real-time detector using 38 layer cascade
6061 features in all layers

[Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv/]

Kristen Grauman

Cascading classifiers for detection

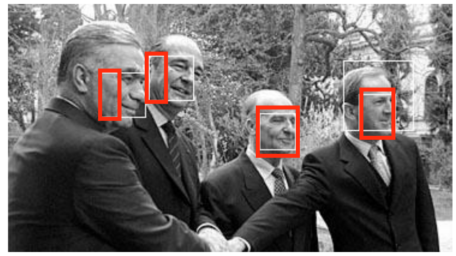


- Form a *cascade* with low false negative rates early on
- Apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative

Kristen Grauman

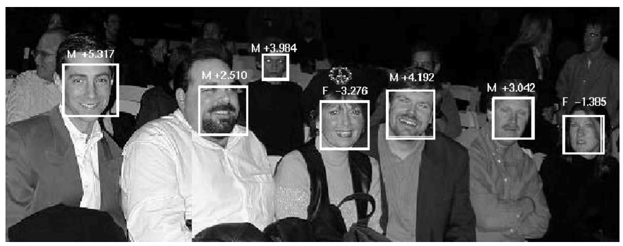
Solving other "Face" Tasks

Facial Feature Localization



Profile Detection

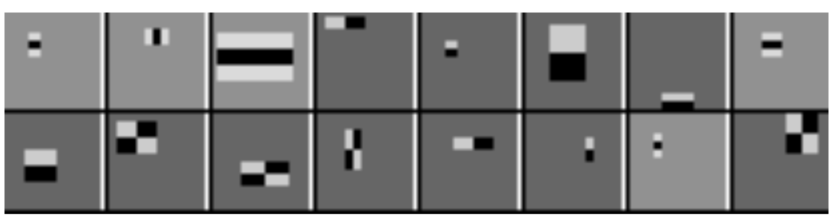
Demographic Analysis

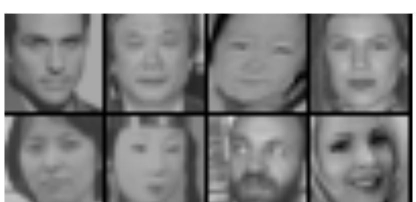


85 Slide credit: Frank Dellaert, Paul Viola, Foryth&Ponce

Face Localization Features

- Learned features reflect the task





Slide credit: Frank Dellaert, Paul Viola, Forsyth&Ponce

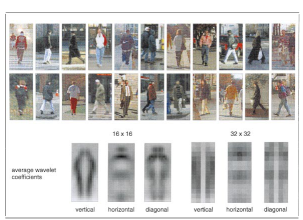
86

Face Profile Detection

87 Slide credit: Frank Dellaert, Paul Viola, Foryth&Ponce

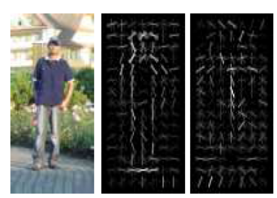
Pedestrian detection

- Detecting upright, walking humans also possible using sliding window's appearance/texture; e.g.,



SVM with Haar wavelets [Papageorgiou & Poggio, IJCV 2000]

Space-time rectangle features [Viola, Jones & Snow, ICCV 2003]

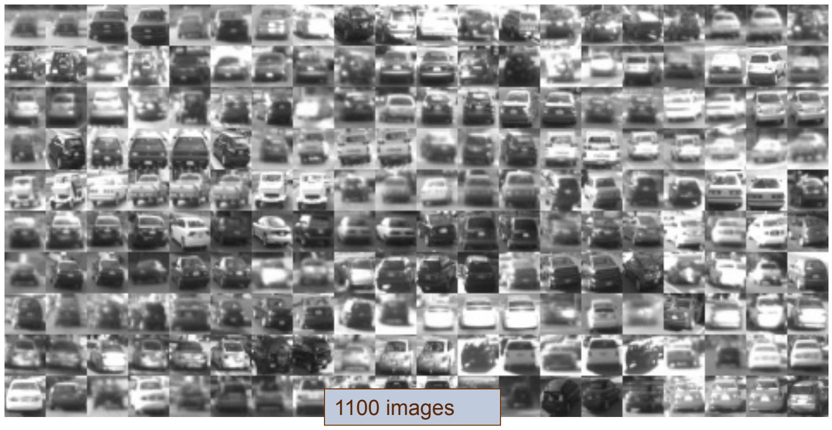


SVM with HoGs [Dalal & Triggs, CVPR 2005]

Kristen Grauman

Finding Cars (DARPA Urban Challenge)

- Hand-labeled images of generic car rear-ends
- Training time: ~5 hours, offline



Credit: Hendrik Dahlkamp

89

Generating even more examples

- Generic classifier finds all cars in recorded video.
- Compute offline and store in database

Credit: Hendrik Dahlkamp

90

Window-based detection: strengths

- Sliding window detection and global appearance descriptors:
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes

Kristen Grauman

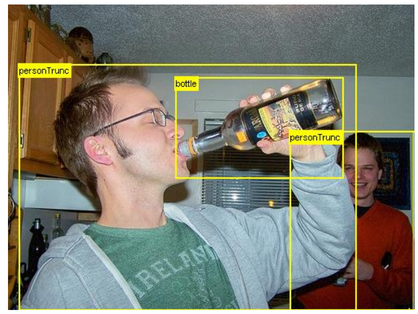
Window-based detection: Limitations

- High computational complexity
  - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
  - If training binary detectors independently, means cost increases linearly with number of classes
- With so many windows, false positive rate better be low

Kristen Grauman

Limitations (continued)

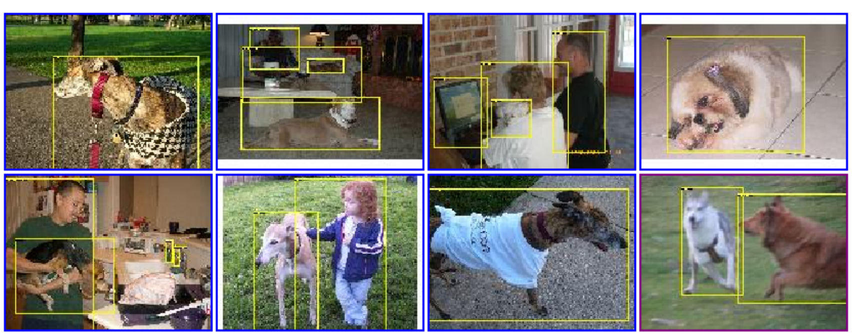
- Not all objects are “box” shaped



Kristen Grauman

Limitations (continued)

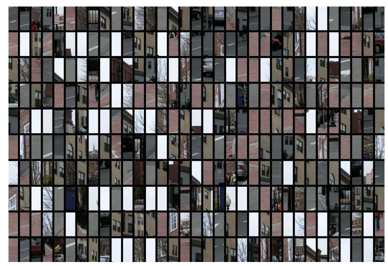
- Non-rigid, deformable objects not captured well with representations assuming a fixed 2d structure; or must assume fixed viewpoint
- Objects with less-regular textures not captured well with holistic appearance-based descriptions



Kristen Grauman

Limitations (continued)

- If considering windows in isolation, context is lost



Sliding window Detector’s view

Figure credit: Derek Hoiem

Limitations (continued)

- In practice, often entails large, cropped training set (expensive)
- Requiring good match to a global appearance description can lead to sensitivity to partial occlusions

Image credit: Adam, Rivlin, & Shimshoni

Summary

- Basic pipeline for window-based detection
  - Model/representation/classifier choice
  - Sliding window and classifier scoring
- Boosting classifiers: general idea
- Viola-Jones face detector
  - Exemplar of basic paradigm
  - Plus key ideas: rectangular features, Adaboost for feature selection, cascade
- Pros and cons of window-based detection

Summary Viola-Jones

- Rectangle features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows

- Many simple features
  - Generalized Haar features (multi-rectangles)
  - Easy and efficient to compute
- Discriminative Learning:
  - finds a small subset for object recognition
  - Uses AdaBoost
- Result: Feature Cascade

98

Summary: Discriminative methods

- Nearest-neighbor and k-nearest-neighbor classifiers
  - L1 distance, χ^2 distance, quadratic distance, Earth Mover's Distance
- Support vector machines
  - Linear classifiers
  - Margin maximization
  - The kernel trick
  - Kernel functions: histogram intersection, generalized Gaussian, pyramid match
  - Multi-class
- Of course, there are many other classifiers out there
  - Neural networks, boosting, decision trees, …

Examples of sliding window based detectors

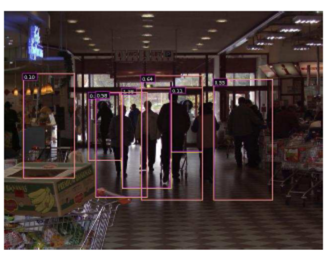
Face detection, Pedestrian detection

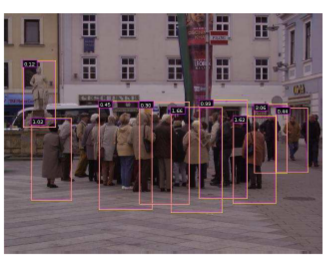
Human detection

- HOG features
- Cue integration
- Ensemble of classifiers
- ROC curve

- Reading: Assigned papers

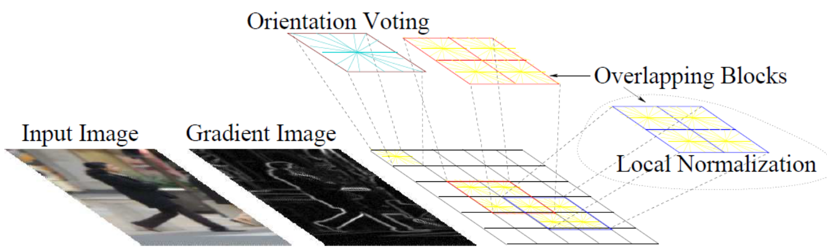
Human detection with HOG

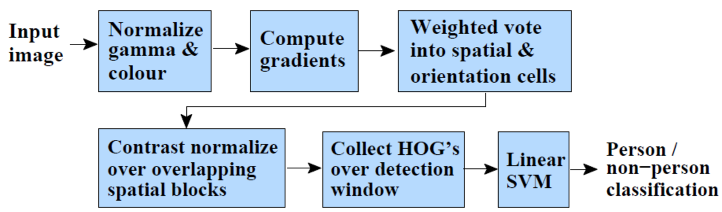




- Histogram of oriented gradients
- Using local gradients to represent positive and negative examples

Histogram of oriented gradients





HOG descriptors

Parameters

- Gradient scale
- Orientation bins
- Percentage of block overlap

Schemes

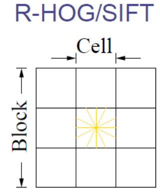
- RGB or Lab, color/gray-space
- Block normalization,

  $L2$-norm, $\mathbf{v} \rightarrow \mathbf{v}/\sqrt{\|\mathbf{v}\|_2^2 + \epsilon^2}$

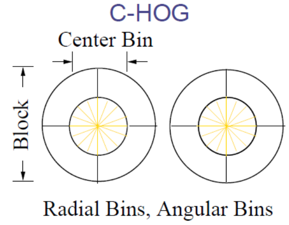
  or

  $L1$-norm, $\mathbf{v} \rightarrow \sqrt{\mathbf{v}/(\|\mathbf{v}\|_1 + \epsilon)}$

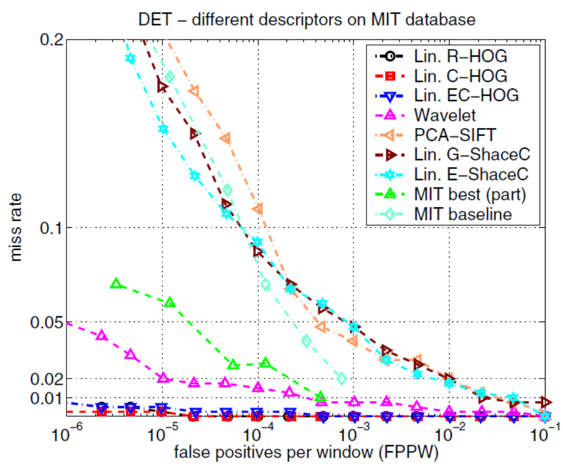
R-HOG/SIFT



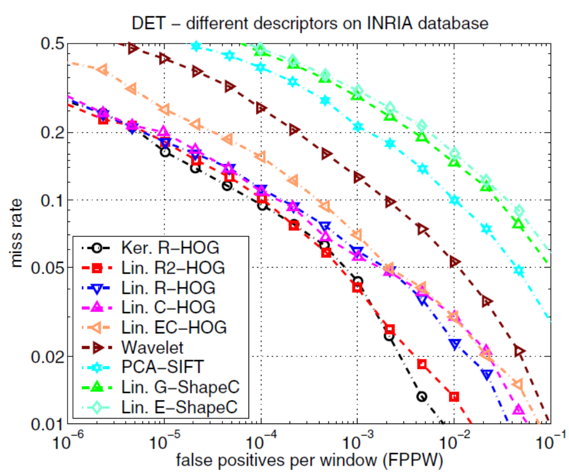
C-HOG



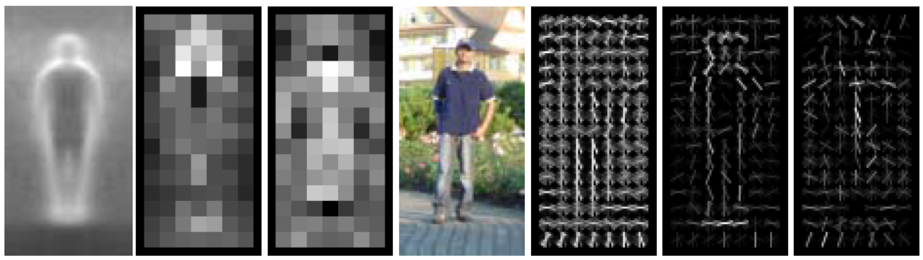
Results with MIT dataset



Results with INRIA dataset



Results



(a) (b) (c) (d) (e) (f) (g)

Figure 6. Our HOG detectors cue mainly on silhouette contours (especially the head, shoulders and feet). The most active blocks are centred on the image background just *outside* the contour. (a) The average gradient image over the training examples. (b) Each "pixel" shows the maximum positive SVM weight in the block centred on the pixel. (c) Likewise for the negative SVM weights. (d) A test image. (e) It's computed R-HOG descriptor. (f,g) The R-HOG descriptor weighted by respectively the positive and the negative SVM weights.

Cal Tech Pedestrian Dataset

	Training			Testing			Height			Properties					
	# pedestrians	# neg. images	# pos. images	# pedestrians	# neg. images	# pos. images	10% quantile	median	90% quantile	color images	per-image ev.	no selec. bias	video seqs.	temporal corr.	occ. labels
MIT[27]	924	–	–	–	–	–	128	128	128	✓					
USC-A[43]	–	–	–	313	–	205	70	98	133		✓				
USC-B[43]	–	–	–	271	–	54	63	90	126		✓				
USC-C[44]	–	–	–	232	–	100	74	108	145		✓	✓			
CVC[14]	1000	6175†	–	–	–	–	46	83	164	✓		✓			
TUD-det[1]	400	–	400	311	–	250	133	218	278	✓	✓				
INRIA[5]	1208	1218	614	566	453	288	139	279	456	✓					
DC[24]	2.4k	15k†	–	1.6k	10k†	–	36	36	36			✓			
ETH[8]	2388	–	499	12k	–	1804	50	90	189	✓	✓	✓	✓		
Caltech	192k	61k	67k	155k	56k	65k	27	48	97	✓	✓	✓	✓	✓	✓

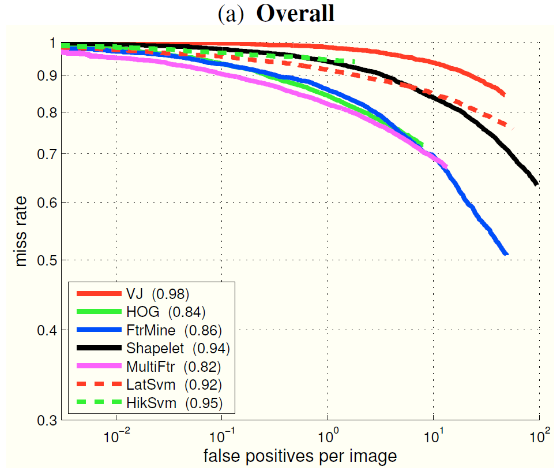
	low-level features	classifier	original im-plementation	trained on INRIA data	per-image evaluation	sec / frame	model height (in pixels)	scale stride	publication
VJ[40]	Haar	AdaBoost		✓		7.0	96	1.05	'04
HOG[5]	HOG	linear SVM	✓	✓		13.3	96	1.05	'05
FtrMine[7]	gen. Haar	AdaBoost	✓	✓		45	96	1.20	'07
Shapelet[30]	gradients	AdaBoost		✓		60.1	96	1.05	'07
MultiFtr[42]	HOG+Haar	AdaBoost	✓	✓	✓	18.9	96	1.05	'08
LatSvm[11]	HOG	latent SVM	✓		✓	6.3	80	1.05	'08
HikSvm[21]	HOG-like	HIK SVM	✓	✓		140	96	1.20	'08

Figure 10. Selected HOG false negatives (left) and high confidence false positives (right) for *near* scale unoccluded pedestrians.

Performance evaluation

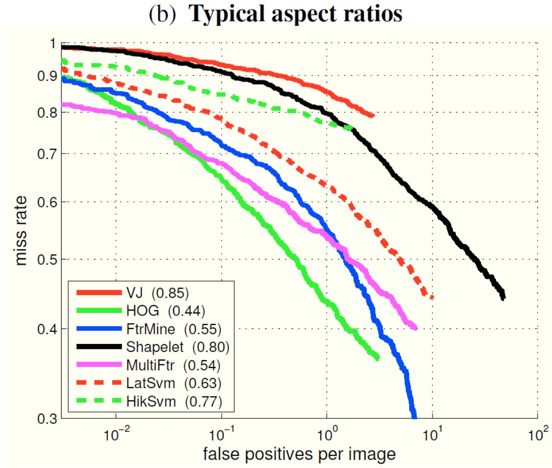
overall & aspect ratio

(a) **Overall**

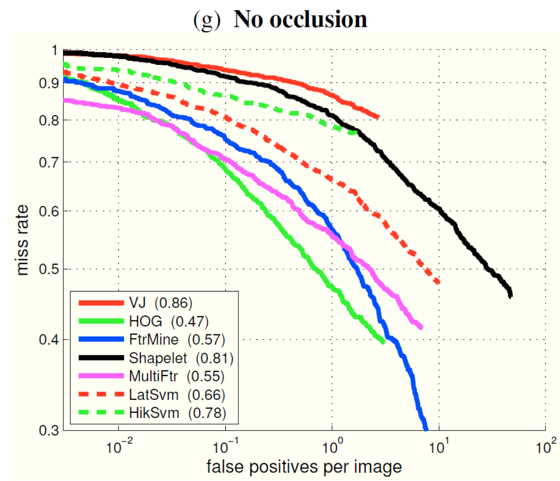


Results (cont' d)

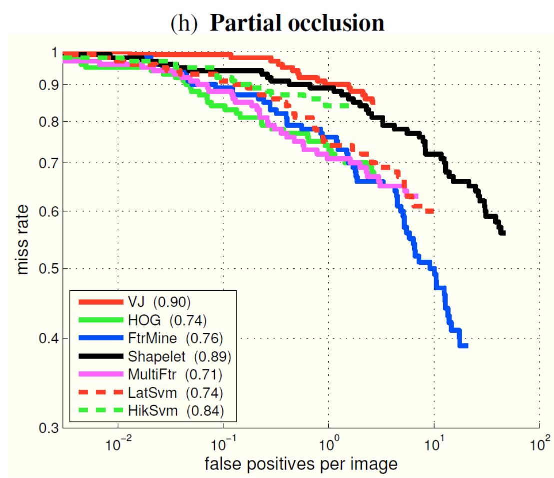
(b) **Typical aspect ratios**



Results (cont' d)

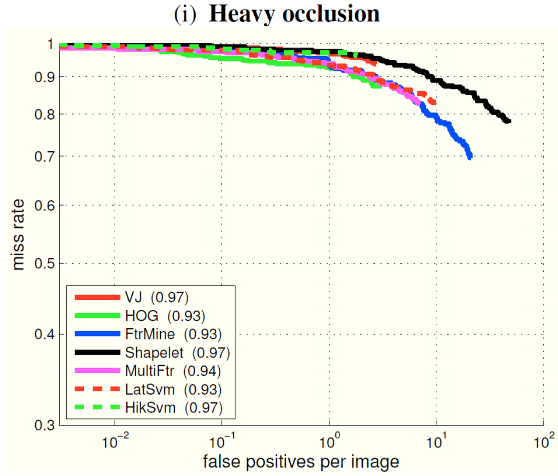


Results (cont' d)



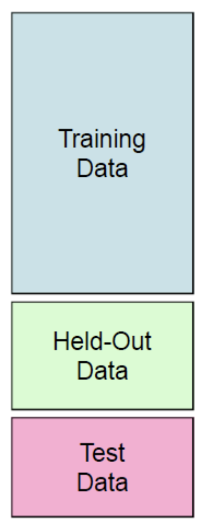
Results (cont' d)

(i) **Heavy occlusion**



Important Concepts

- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set
- Features: attribute-value pairs which characterize each x
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - Compute accuracy of test set
  - Very important: never "peek" at the test set!
- Evaluation
  - Accuracy: fraction of instances predicted correctly
- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well



Slide credit: Dan Klein

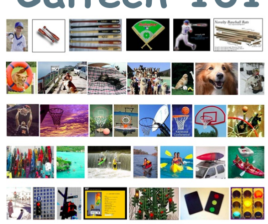
Representation Alternatives

Sparse Coding Introduction

Adopted from: Deep Learning Methods for Vision, CVPR 2012 Tutorial

Relentless research on visual recognition

Caltech 101



80 Million Tiny Images

PASCAL VOC

ImageNet

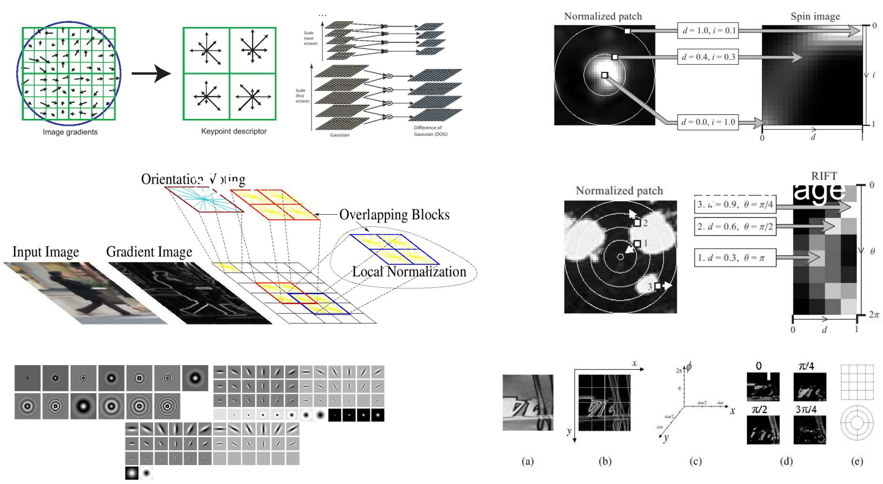
The pipeline of machine visual perception

Most Efforts in Machine Learning

Low-level sensing → Pre-processing → Feature extract. → Feature selection → Inference: prediction, recognition

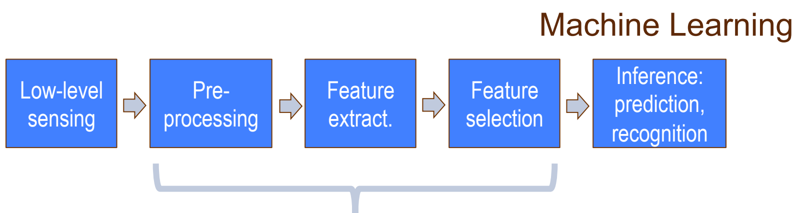
- Most critical for accuracy
- Account for most of the computation for testing
- Most time-consuming in development cycle
- Often hand-craft in practice

Computer vision features



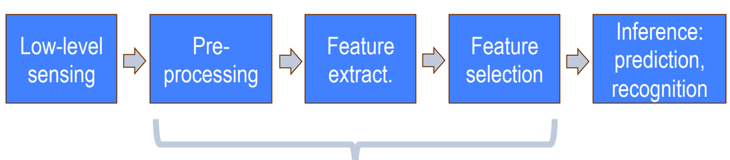
Slide Credit: Andrew Ng

Learning features from data



Feature Learning:
instead of design features,
let's design feature learners

Learning features from data via sparse coding



Sparse coding offers an effective building block to learn useful features

Outline

1. Sparse coding for image classification
2. Understanding sparse coding
3. Hierarchical sparse coding
4. Other topics: e.g. structured model, scale-up, discriminative training
5. Summary

9/10/13 132

"BoW representation + SPM" Paradigm - I

Bag-of-visual-words representation (BoW) based on VQ coding

9/10/13 133

Figure credit: Fei-Fei Li

“BoW representation + SPM” Paradigm - II

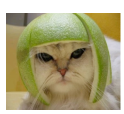
Spatial pyramid matching:
pooling in different scales and locations

9/10/13 134

Figure credit: Svetlana Lazebnik

Image Classification using “BoW+SPM”

Image Classification



Dense SIFT

↓

VQ Coding

↓

Spatial Pooling

↓

Classifier

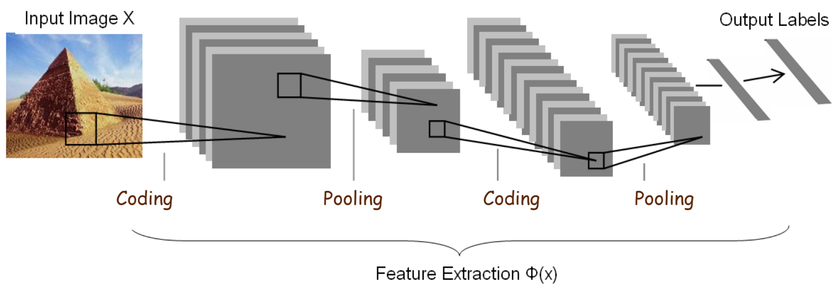
$$\left[x^{(1)}, x^{(2)}, \ldots, x^{(m)}\right] \in \mathbb{R}^{128}$$

$$\left[a^{(1)}, a^{(2)}, \ldots, a^{(m)}\right] \in \mathbb{R}^{k}$$

$$a = \sum_{i=1}^{m} v_i a^{(i)}$$

9/10/13 135

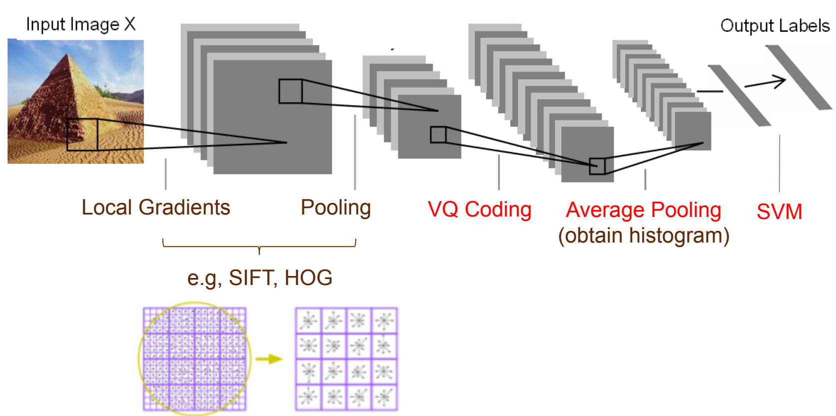
The Architecture of "Coding + Pooling"



• e.g., convolutional neural net, HMAX, BoW, …

136

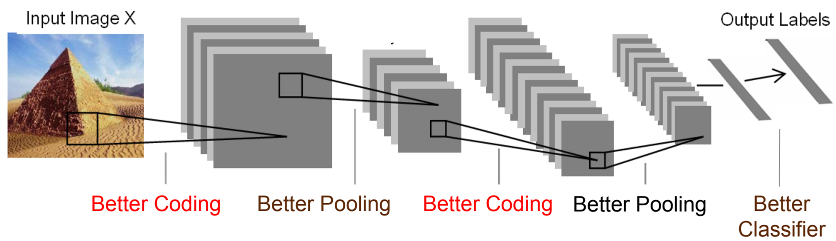
"BoW+SPM" has two coding+pooling layers



SIFT feature itself follows a coding+pooling operation

137

Develop better coding methods



- Coding: nonlinear mapping data into another feature space
- Better coding methods: sparse coding, RBMs, auto-encoders

What is sparse coding

Sparse coding (Olshausen & Field,1996). Originally developed to explain early visual processing in the brain (edge detection).

Training: given a set of random patches x, learning a dictionary of bases [$\Phi_1, \Phi_2, \ldots$]

Coding: for data vector x, solve LASSO to find the sparse coefficient vector a

$$\min_{a,\phi} \sum_{i=1}^{m} \left\| x_i - \sum_{j=1}^{k} a_{i,j}\phi_j \right\|^2 + \lambda \sum_{i=1}^{m}\sum_{j=1}^{k} |a_{i,j}|$$

Sparse coding: training time

Input: Images $x_1, x_2, \ldots, x_m$ (each in R^d)
Learn: Dictionary of bases $\phi_1, \phi_2, \ldots, \phi_k$ (also R^d).

$$\min_{a,\phi} \sum_{i=1}^{m} \left\| x_i - \sum_{j=1}^{k} a_{i,j}\phi_j \right\|^2 + \lambda \sum_{i=1}^{m}\sum_{j=1}^{k} |a_{i,j}|$$

Alternating optimization:

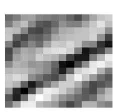
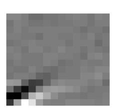
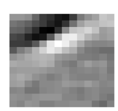
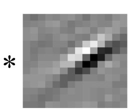
1. Fix dictionary $\phi_1, \phi_2, \ldots, \phi_k$, optimize a (a standard LASSO problem)

2. Fix activations a, optimize dictionary $\phi_1, \phi_2, \ldots, \phi_k$ (a convex QP problem)

Sparse coding: testing time

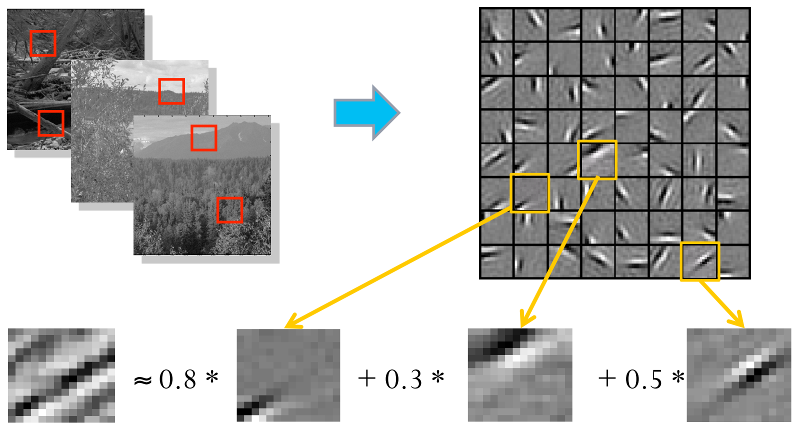
Input: Novel image patch x (in R^d) and previously learned ϕ_i's
Output: Representation $[a_{i,1}, a_{i,2}, \ldots, a_{i,\kappa}]$ of image patch x_i.

$$\min_{a} \sum_{i=1}^{m} \left\| x_i - \sum_{j=1}^{k} a_{i,j}\phi_j \right\|^2 + \lambda \sum_{i=1}^{m}\sum_{j=1}^{k} |a_{i,j}|$$

 $\approx 0.8 *$  $+ 0.3 *$  $+ 0.5 *$ 

Represent x_i as: a_i = [0, 0, …, 0, **0.8**, 0, …, 0, **0.3**, 0, …, 0, **0.5**, …]

Sparse coding illustration



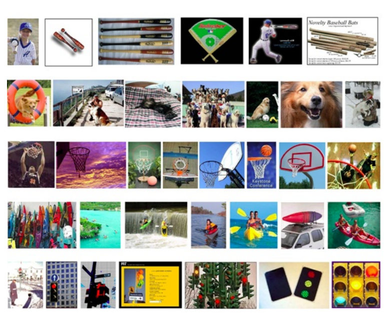
$[a_1, \ldots, a_{64}]$ = [0, 0, …, 0, **0.8**, 0, …, 0, **0.3**, 0, …, 0, **0.5**, 0]
(feature representation)

Slide credit: Andrew Ng

Compact & easily interpretable

Classification Result on Caltech 101

9K images, 101 classes



64%

SIFT VQ +
Nonlinear SVM

~50%

Pixel Sparse Coding
+ Linear SVM

9/10/13 144