Object recognition

Methods for classification and image representation

Credits

- Slides by Pete Barnum
- Slides by Fei-Fei Li

- Paul Viola, Michael Jones, Robust Real-time Object Detection, IJCV 04
- Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05
- Kristen Grauman, Gregory Shakhnarovich, and Trevor Darrell, Virtual Visual Hulls: Example-Based 3D Shape Inference from Silhouettes
- S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories.
- Yoav Freund Robert E. Schapire, A Short Introduction to Boosting

Object recognition

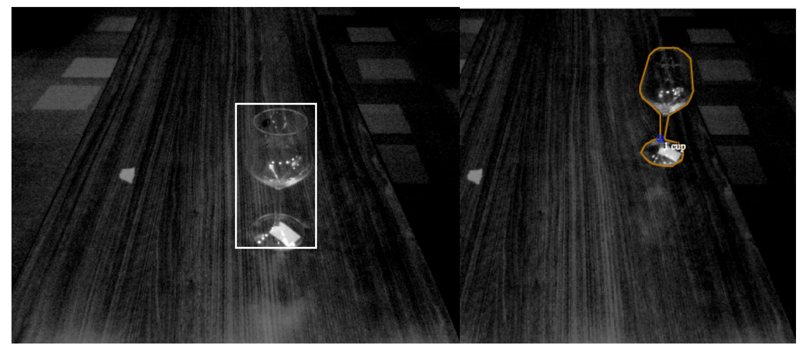
- What is it?
  - Instance
  - Category
  - Something with a tail
- Where is it?
  - Localization
  - Segmentation
- How many are there?

(CC) By Paul Gouden

(CC) By Peter Hellberg

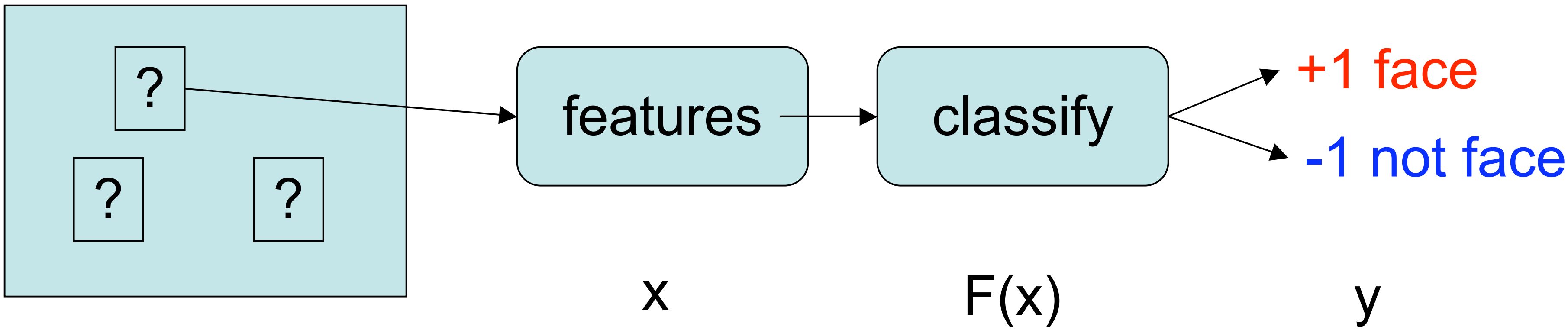
Object recognition

- What is it?
  - Instance
  - Category
  - Something with a tail
- Where is it?
  - Localization
  - Segmentation
- How many are there?



(CC) By Dunechaser

Face detection



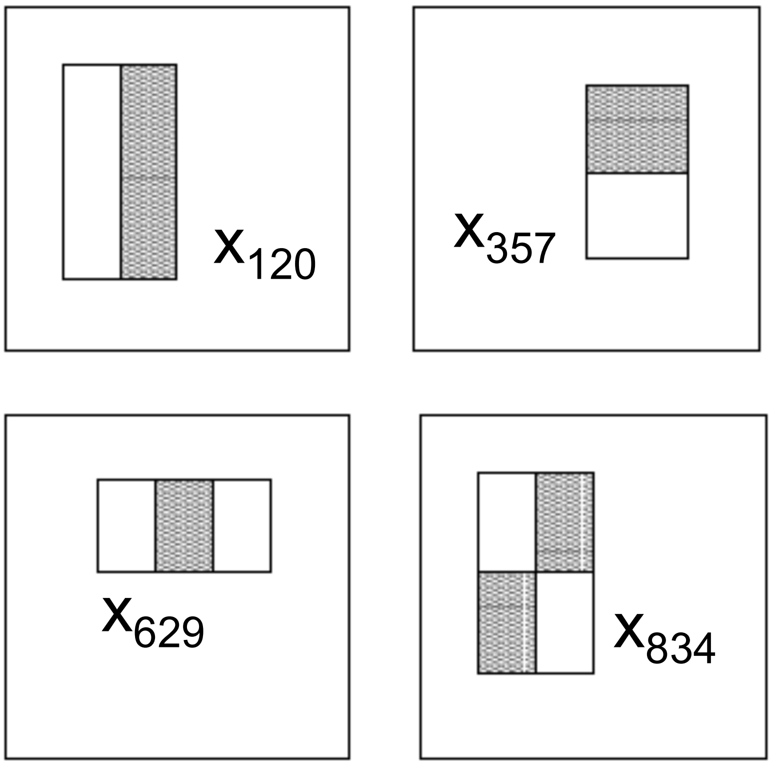
- We slide a window over the image
- Extract features for each window
- Classify each window into face/non-face

What is a face?

- Eyes are dark (eyebrows+shadows)
- Cheeks and forehead are bright.
- Nose is bright

Paul Viola, Michael Jones, Robust Real-time Object Detection, IJCV 04

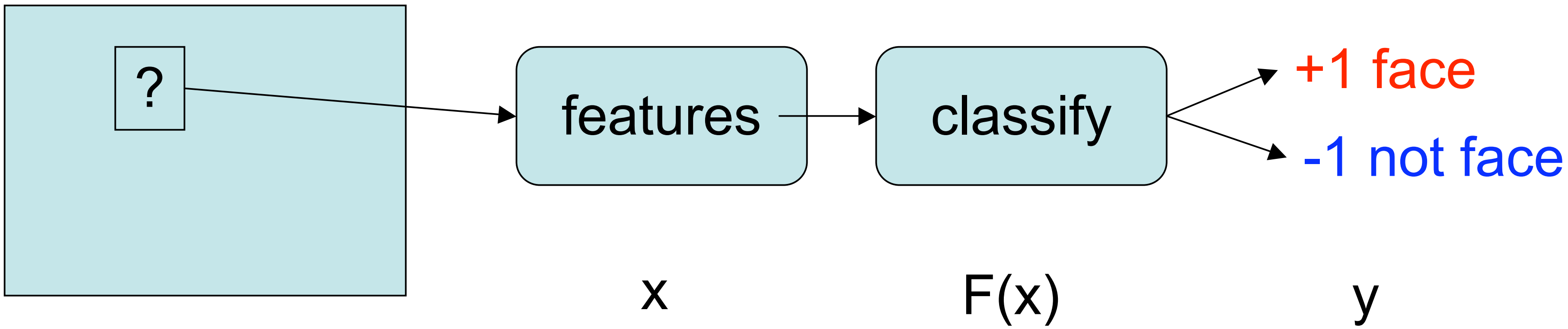
Basic feature extraction



- Information type:
  - intensity
- Sum over:
  - gray and white rectangles
- Output: gray-white
- Separate output value for
  - Each type
  - Each scale
  - Each position in the window
- $FEX(im)=x=[x_1,x_2,\ldots\ldots.,x_n]$

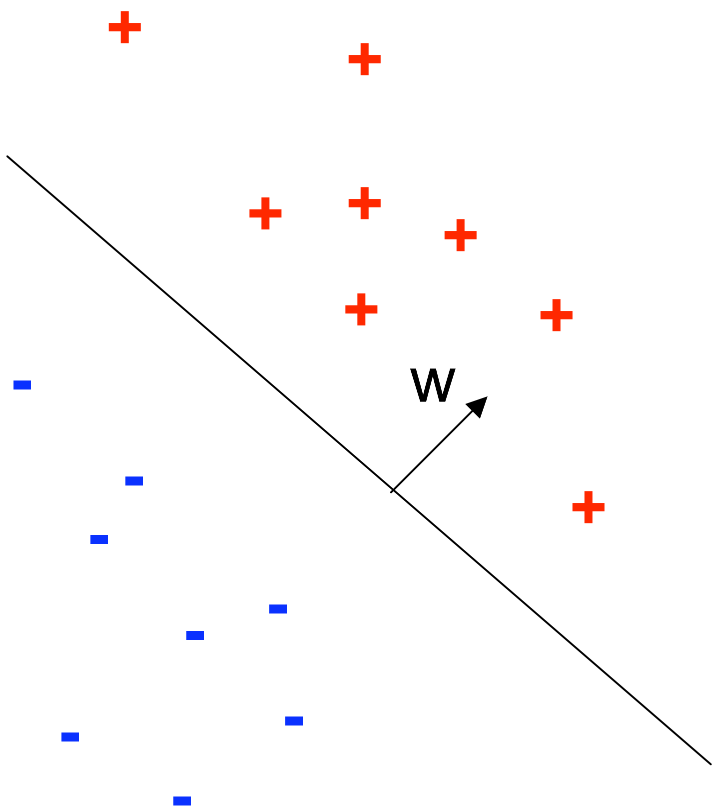
Paul Viola, Michael Jones, Robust Real-time Object Detection, IJCV 04

Face detection



- We slide a window over the image
- Extract features for each window
- Classify each window into face/non-face

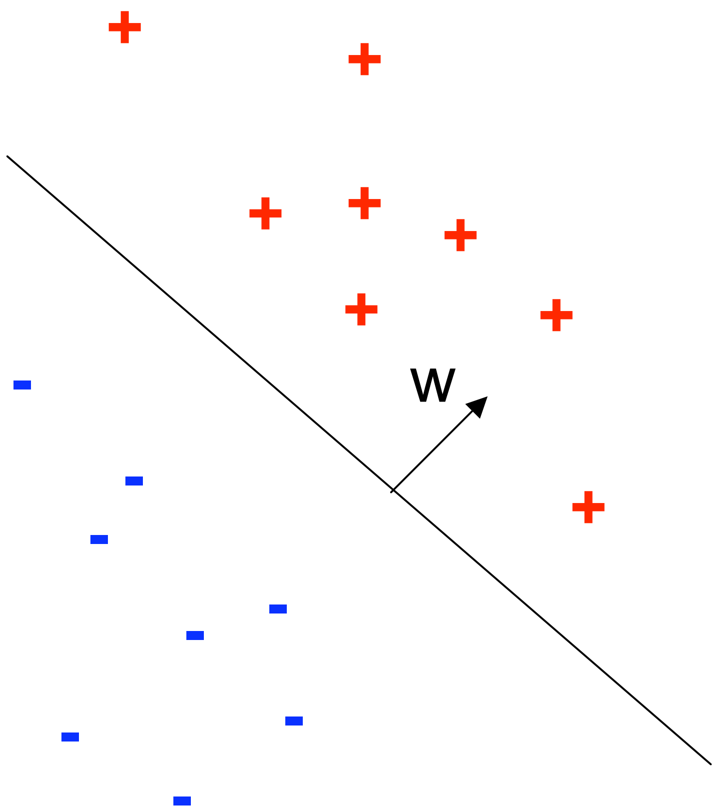
Classification



- Examples are points in R^n
- Positives are separated from negatives by the hyperplane **w**

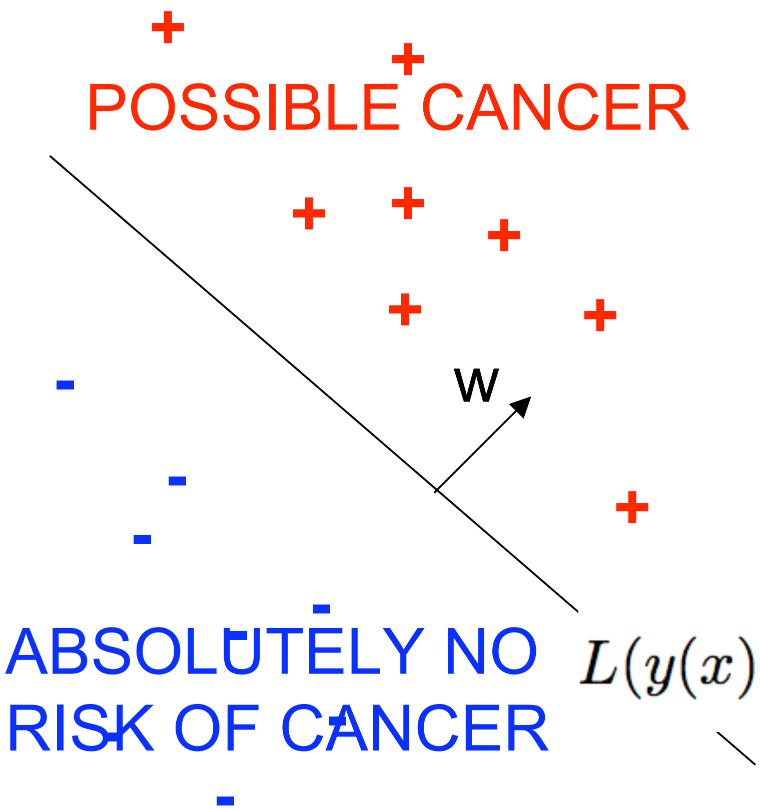
- $y=\text{sign}(w^T x-b)$

Classification



- $x \in R^n$ - data points
- $P(x)$ - distribution of the data
- $y(x)$ - true value of y for each x
- F - decision function:
  $y=F(x, \theta)$
- θ - parameters of F,
  e.g. $\theta=(w,b)$
- We want F that makes few mistakes

Loss function



- Our decision may have severe implications

- L(y(x),F(x, θ)) - loss function
  How much we pay for predicting F(x,θ), when the true value is y(x)

- Classification error:

$$L(y(x), F(x,\theta)) = \begin{cases} 0, & y(x) = sign(w^T x - b) \\ 1, & otherwise \end{cases}$$

- Hinge loss

$$L(y(x), F(x,\theta)) = max(0, 1 - y(x)F(x,\theta))$$

Learning

- Total loss shows how good a function (F, θ) is:

$$L(f) = \int_x L(y, F(x))P(x)dx$$

- Learning is to find a function to minimize the loss:

$$(F, \theta) = \arg\min_{F,\theta} \int_x L(y, F(x, \theta))P(x)dx$$

- How can we see all possible x?

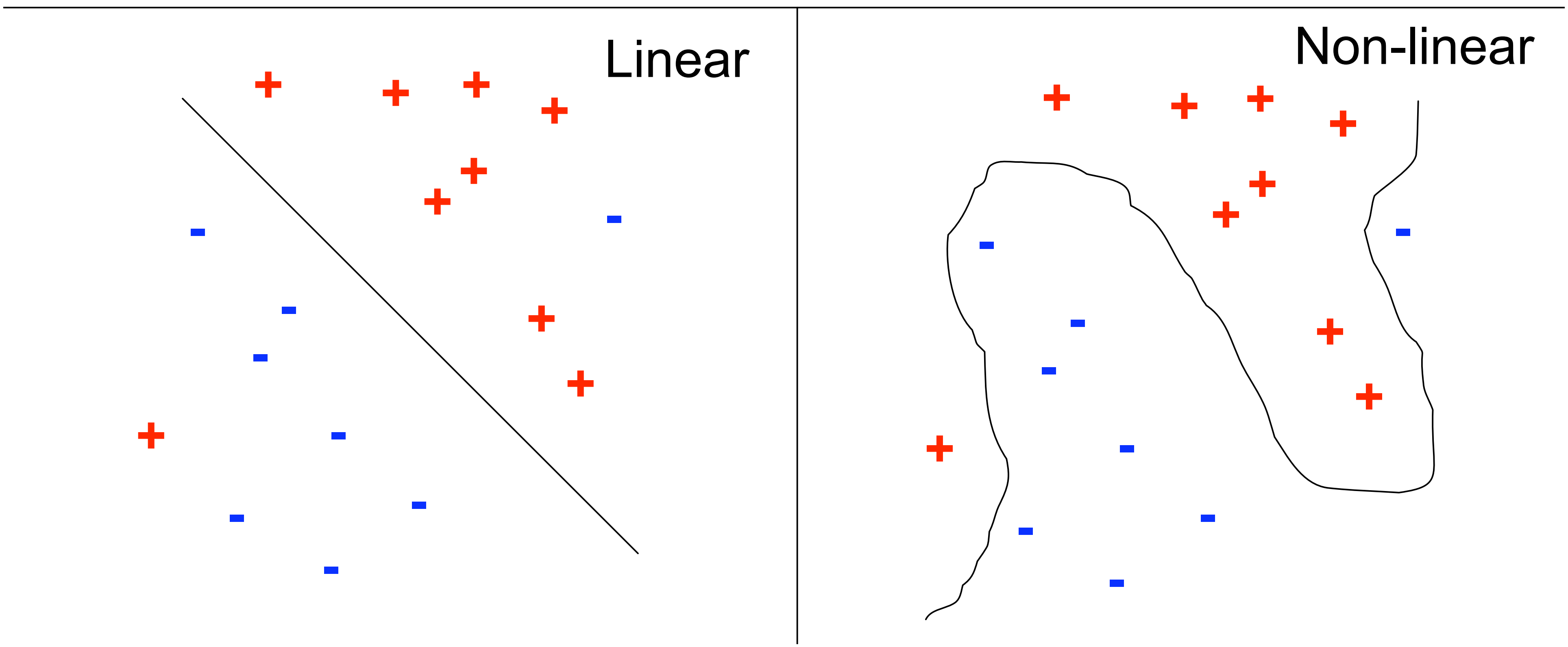
Datasets

- Dataset is a finite sample $\{x_i\}$ from $P(x)$
- Dataset has labels $\{(x_i, y_i)\}$
- Datasets today are big to ensure the sampling is fair

	#images	#classes	#instances
Caltech 256	30608	256	30608
Pascal VOC	4340	20	10363
LabelMe	176975	???	414687

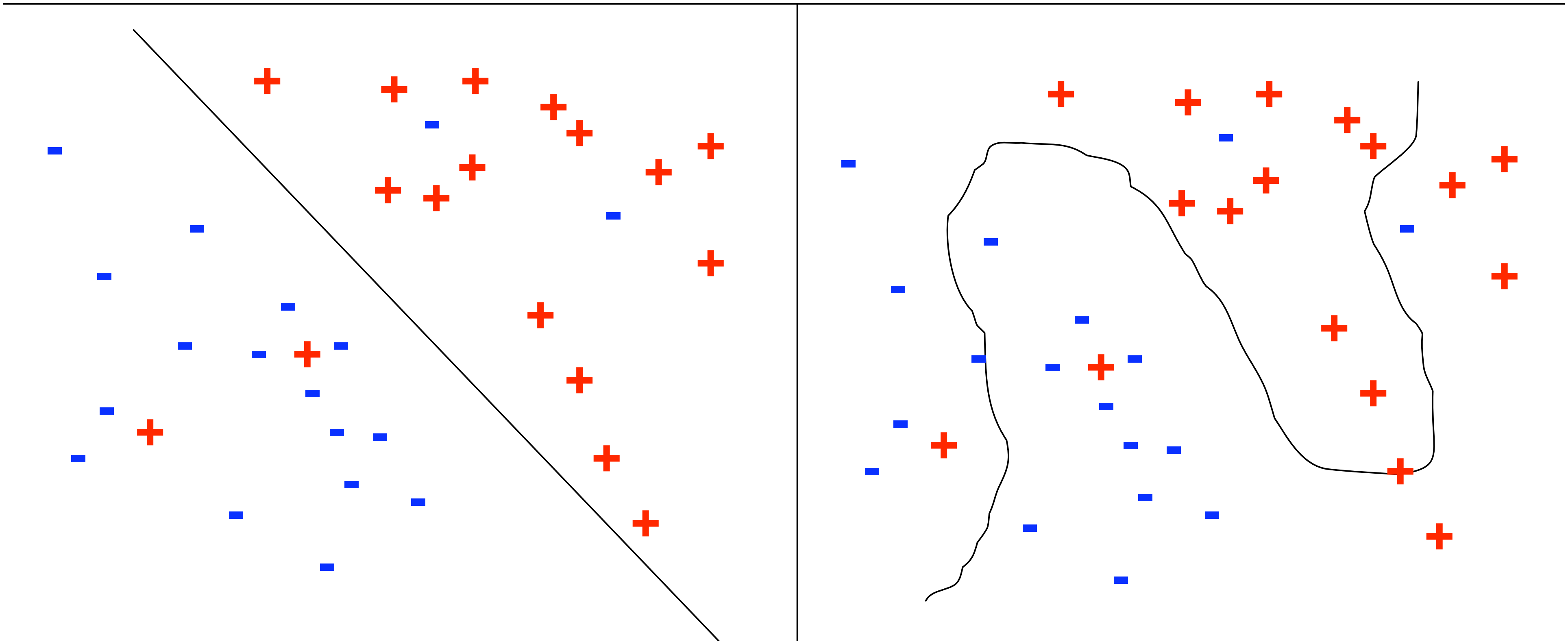
Overfitting

- A simple dataset.
- Two models



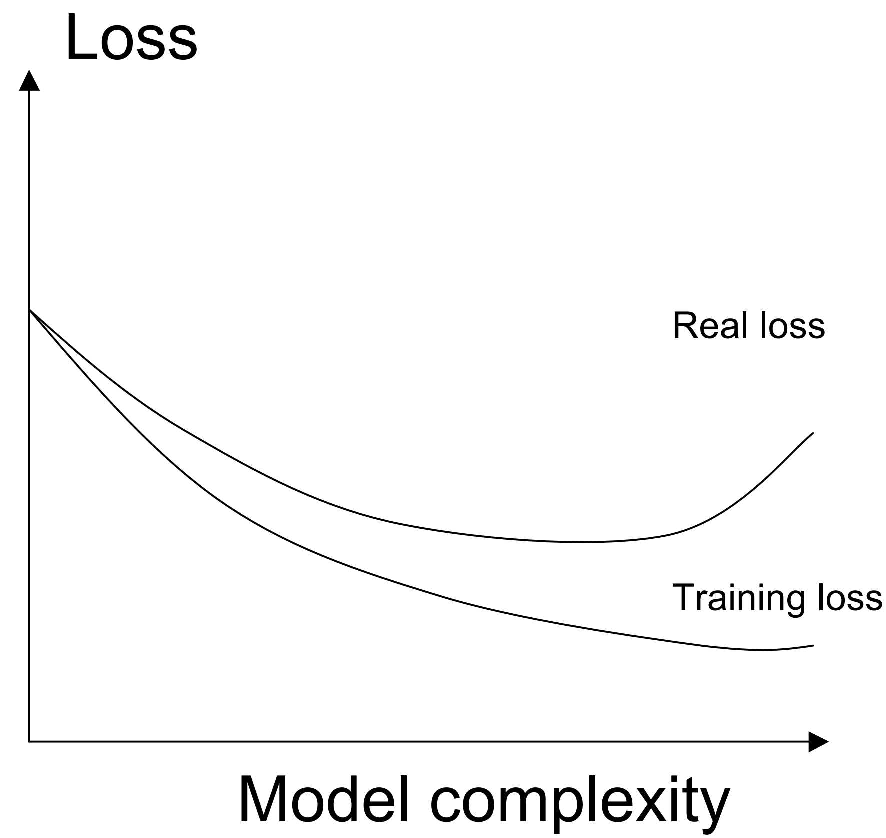
Overfitting

- Let's get more data.
- Simple model has better generalization.



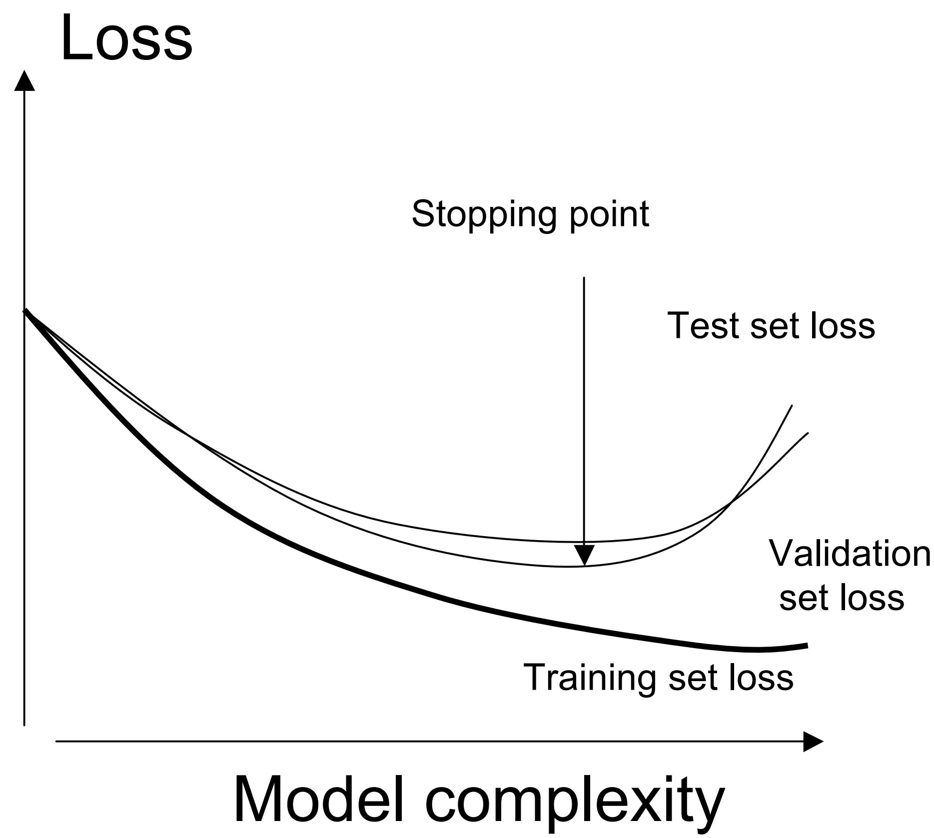
Overfitting

- As complexity increases, the model overfits the data
- Training loss decreases
- Real loss increases
- We need to penalize model complexity
  = to regularize



Overfitting

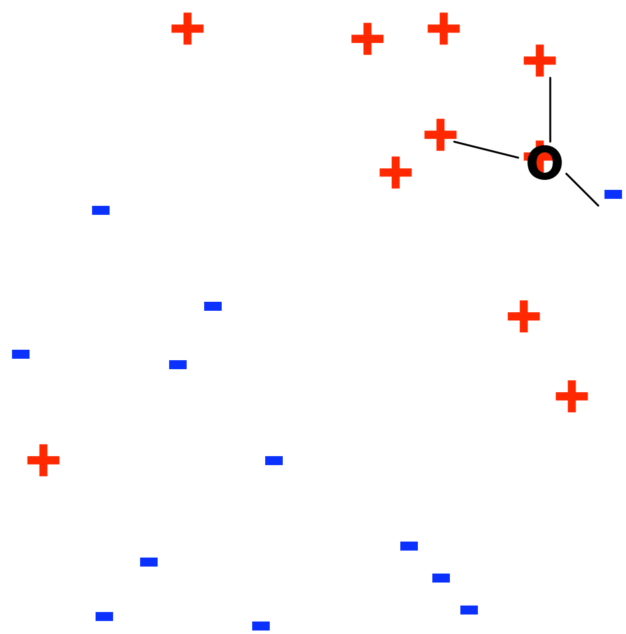
- Split the dataset
  - Training set
  - Validation set
  - Test set
- Use training set to **optimize** model parameters
- Use validation test to **choose** the best model
- Use test set only to **measure** the expected loss



Classification methods

- K Nearest Neighbors
- Decision Trees
- Linear SVMs
- Kernel SVMs
- Boosted classifiers

K Nearest Neighbors

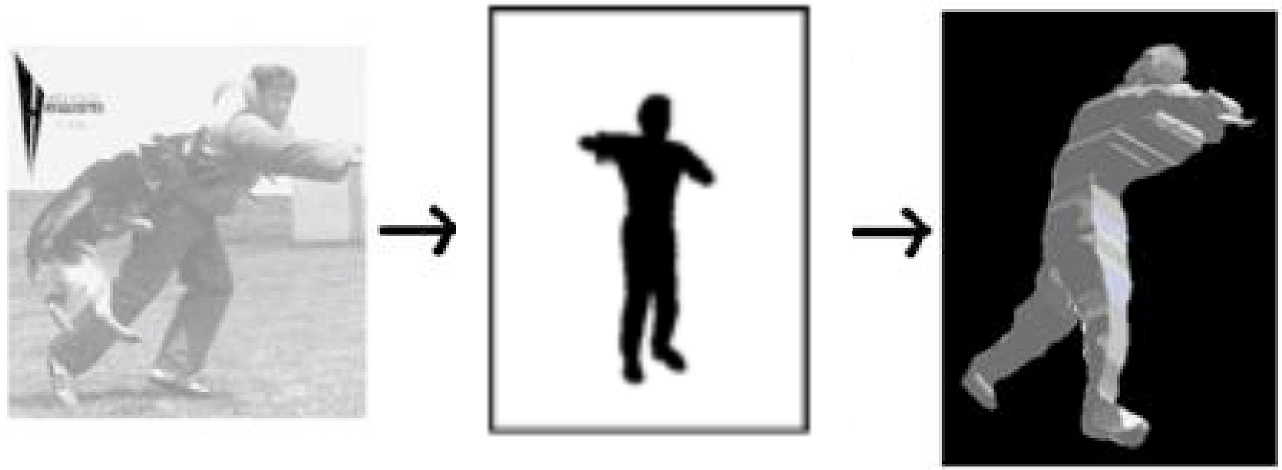


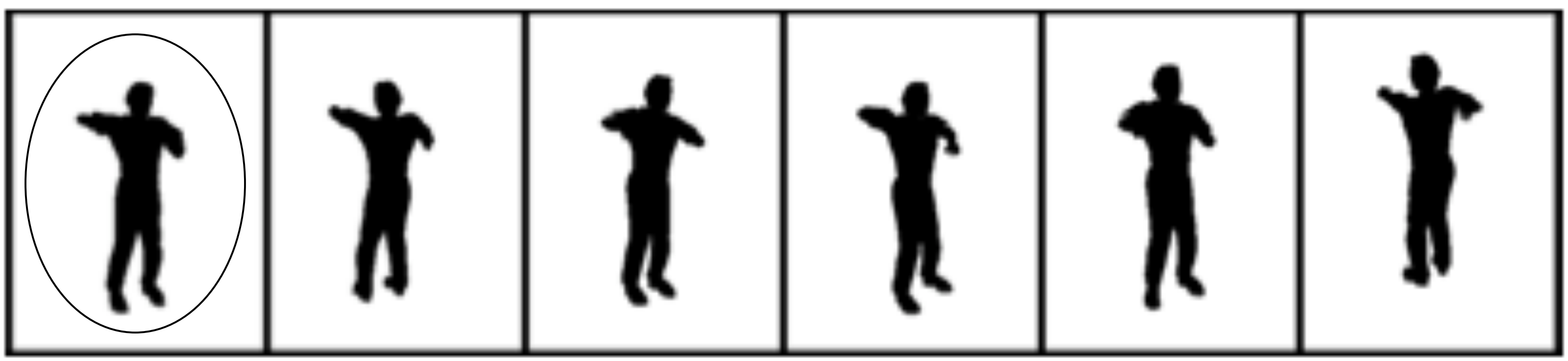
- Memorize all training data
- Find K closest points to the query
- The neighbors vote for the label:

  Vote(+)=2

  Vote(–)=1

K-Nearest Neighbors



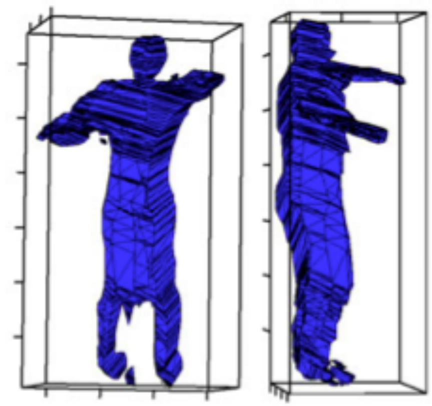


Nearest Neighbors (silhouettes)

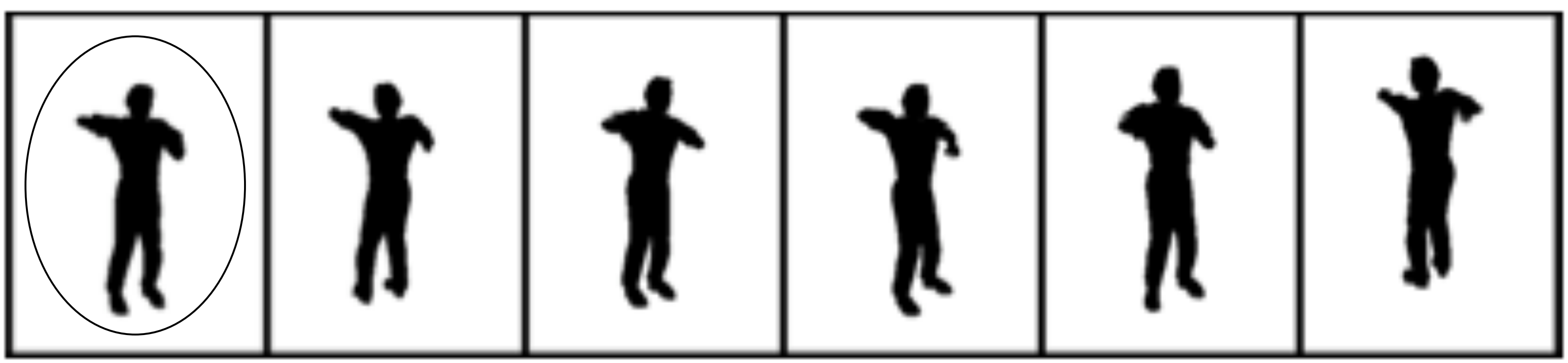
Kristen Grauman, Gregory Shakhnarovich, and Trevor Darrell,
Virtual Visual Hulls: Example-Based 3D Shape Inference from Silhouettes

K-Nearest Neighbors

Silhouettes from other views

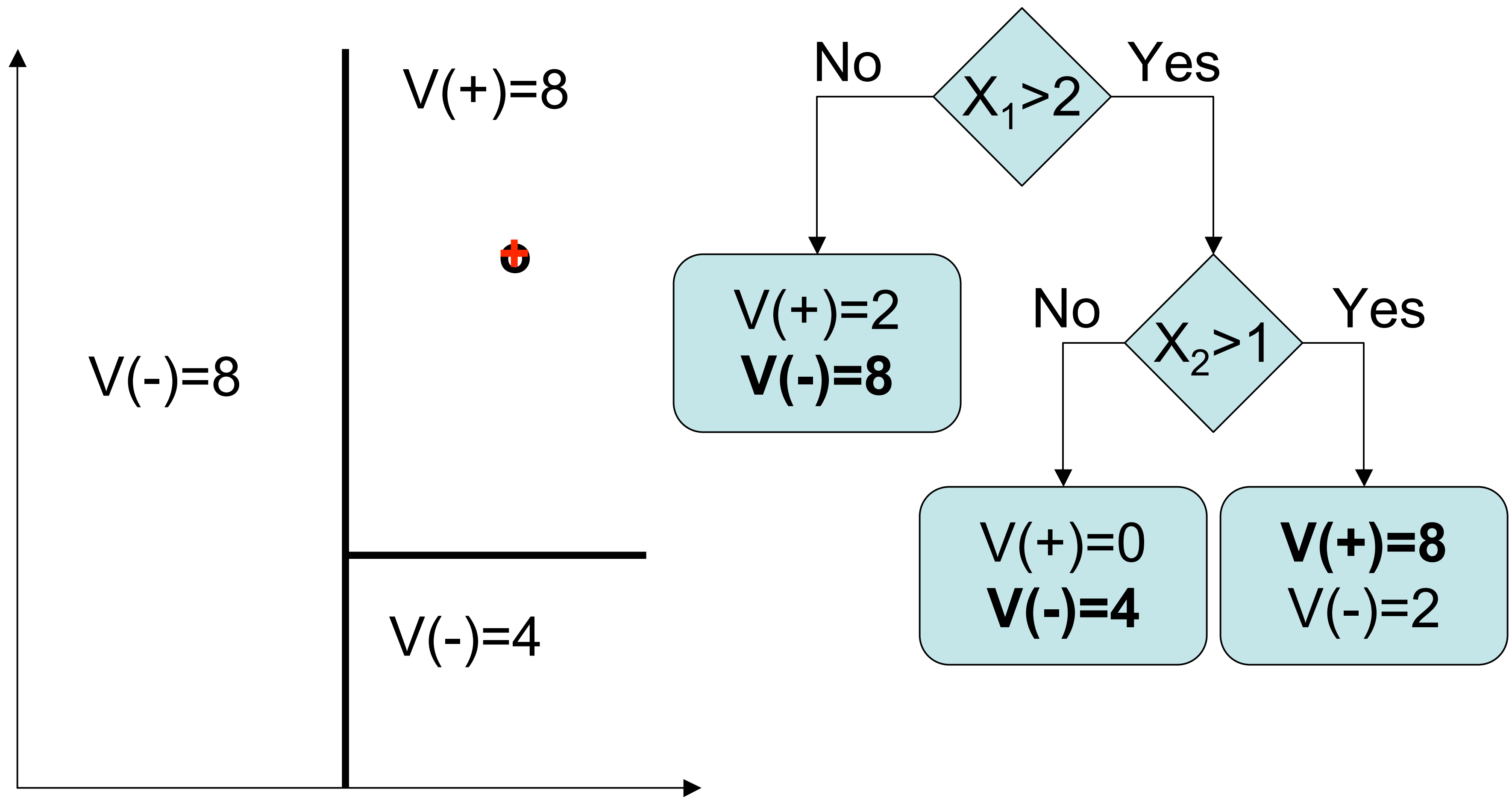


3D Visual hull

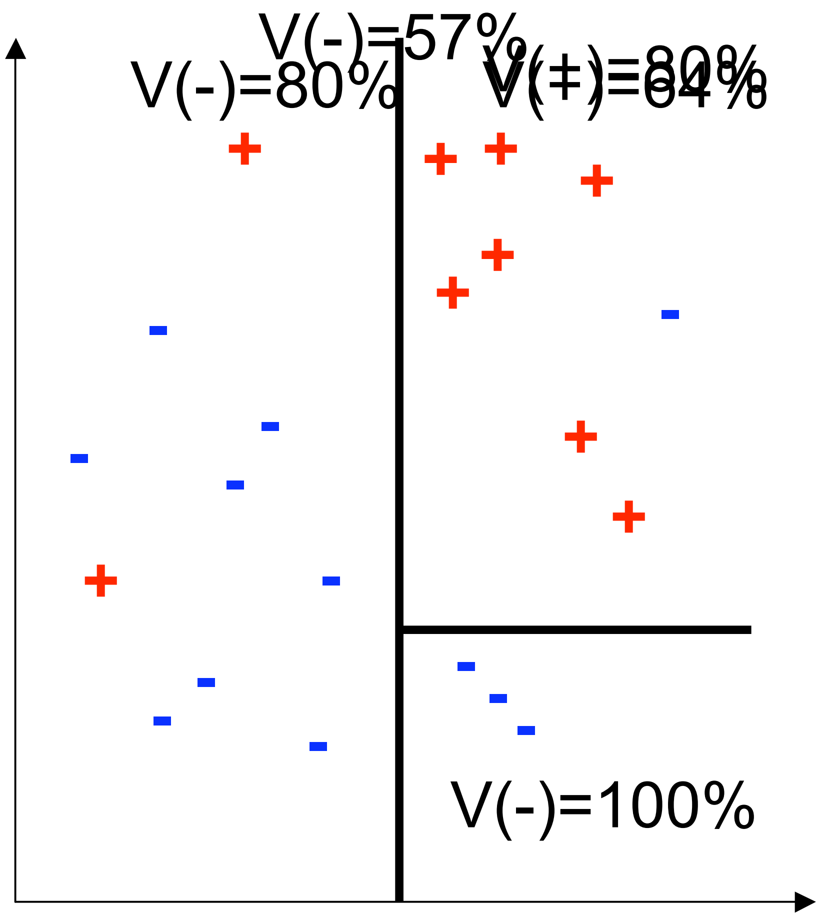


Kristen Grauman, Gregory Shakhnarovich, and Trevor Darrell,
Virtual Visual Hulls: Example-Based 3D Shape Inference from Silhouettes

Decision tree



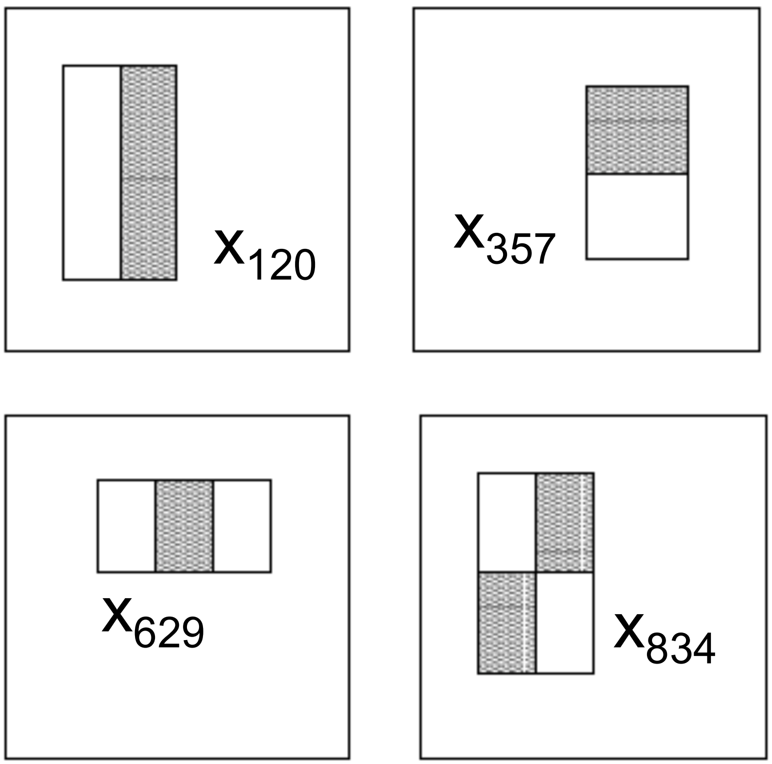
Decision Tree Training



- Partition data into pure chunks

- Find a good rule
- Split the training data
  - Build left tree
  - Build right tree
- Count the examples in the leaves to get the votes: V(+), V(-)
- Stop when
  - Purity is high
  - Data size is small
  - At fixed level

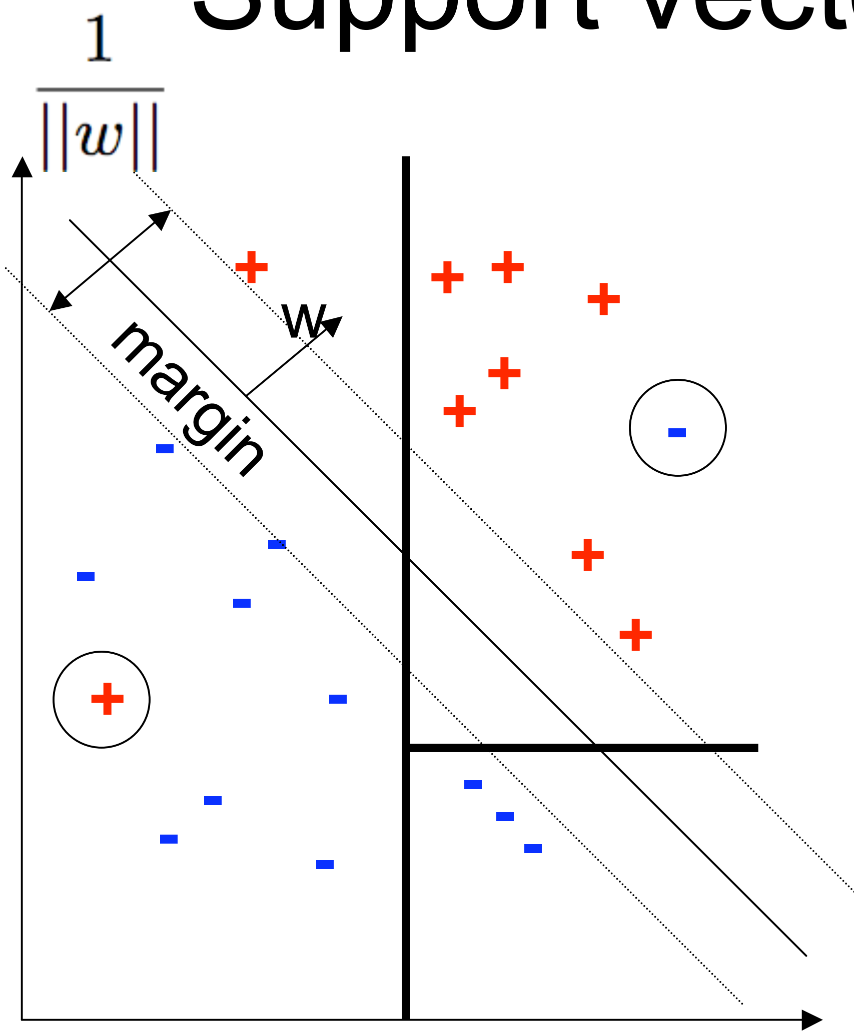
Decision trees



- Stump:
  - 1 root
  - 2 leaves
- If $x_i > a$
  then positive
  else negative
- Very simple
- "Weak classifier"

Paul Viola, Michael Jones, Robust Real-time Object Detection, IJCV 04

Support vector machines



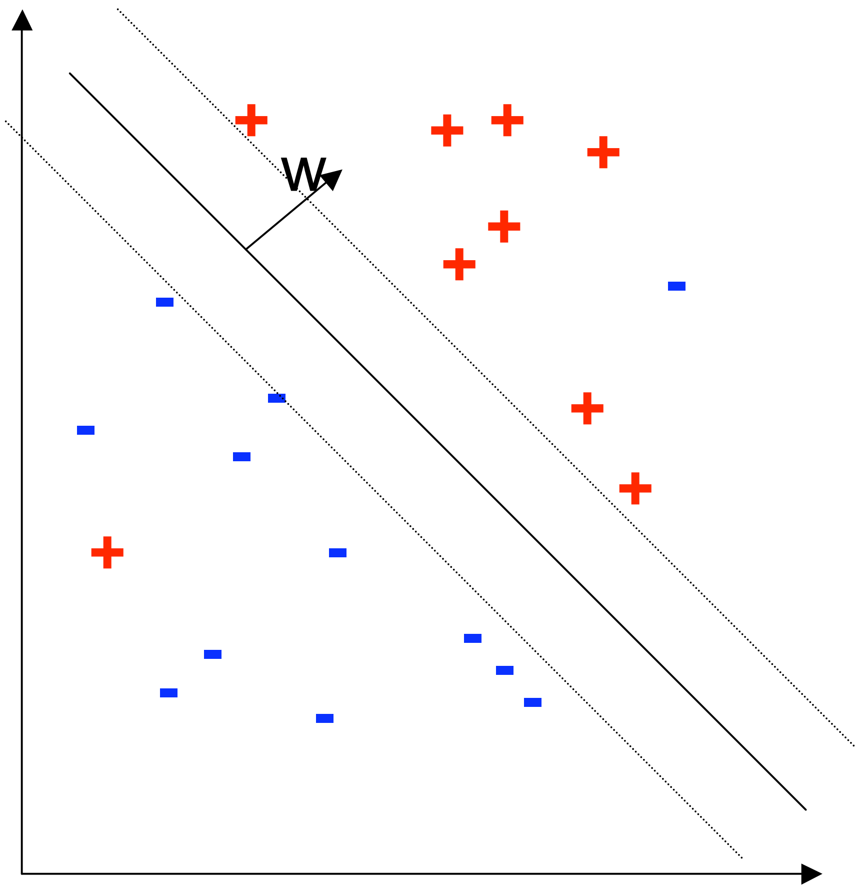
- Simple decision
- Good classification
- Good generalization

$$\min_{w,\xi} \frac{||w||^2}{2} + C \sum_j \xi_j$$

$$y_j \, w^T x_j \geq 1 - \xi_j$$

$$\xi_j \geq 0$$

Support vector machines



$$\min_{w,\xi} \frac{||w||^2}{2} + C \sum_j \xi_j$$

$$y_j\, w^T x_j \geq 1 - \xi_j$$

$$\xi_j \geq 0$$

Support vectors:

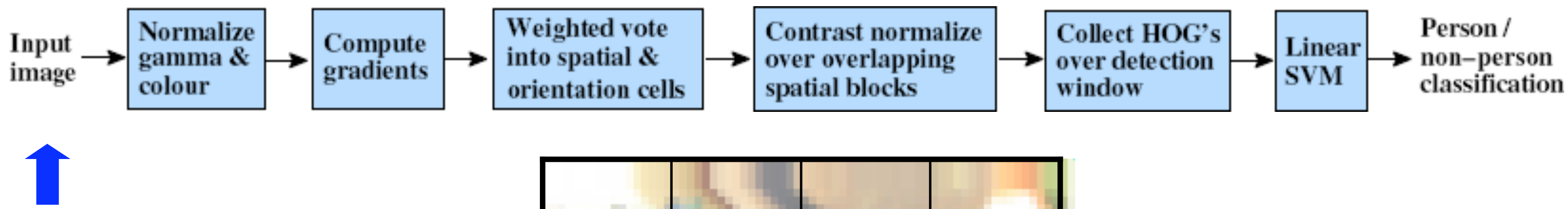
$$w = \sum_{x_i - s.v.} \alpha_i x_i$$

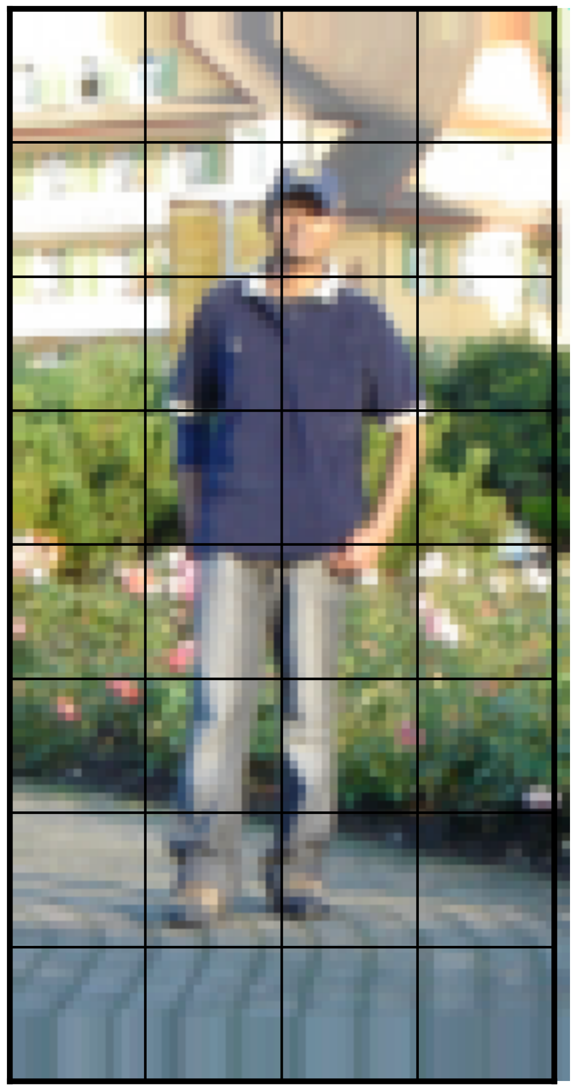
How do I solve the problem?

- It's a convex optimization problem
  - Can solve in Matlab (don't)
- Download from the web
  - SMO: Sequential Minimal Optimization
  - SVM-Light http://svmlight.joachims.org/
  - LibSVM http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - LibLinear http://www.csie.ntu.edu.tw/~cjlin/liblinear/

  - SVM-Perf http://svmlight.joachims.org/
  - Pegasos http://ttic.uchicago.edu/~shai/

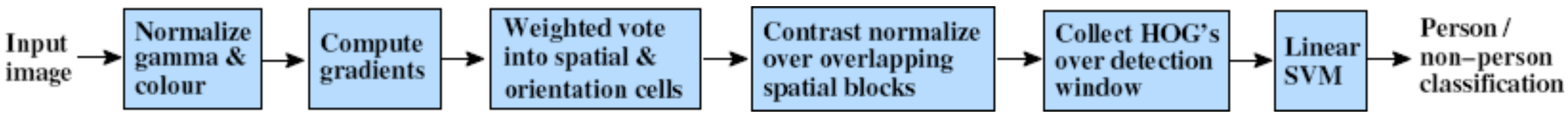
Linear SVM for pedestrian detection





Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



-1	0	1

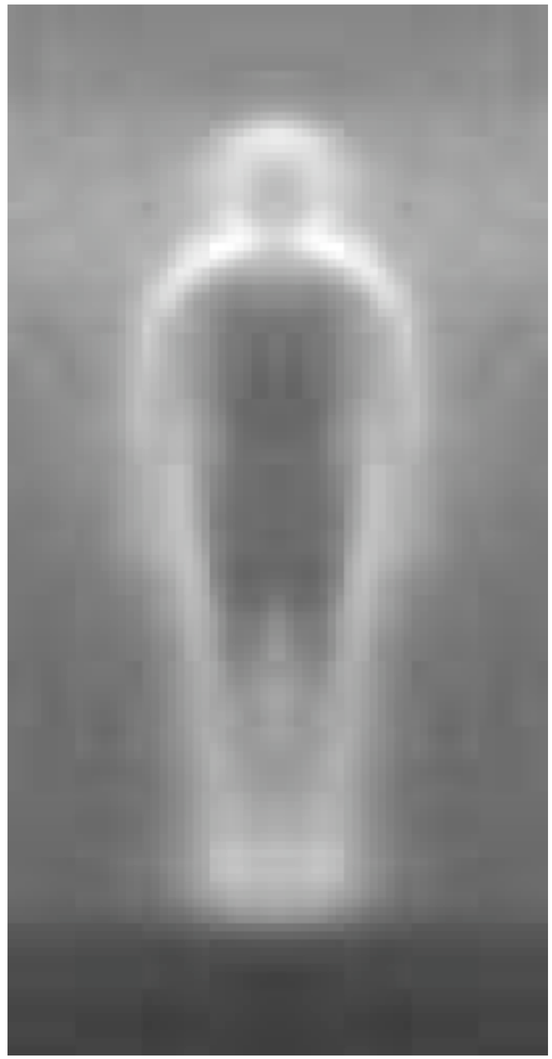
centered

-1	1

uncentered

1	-8	0	8	-1

cubic-corrected



0	1
-1	0

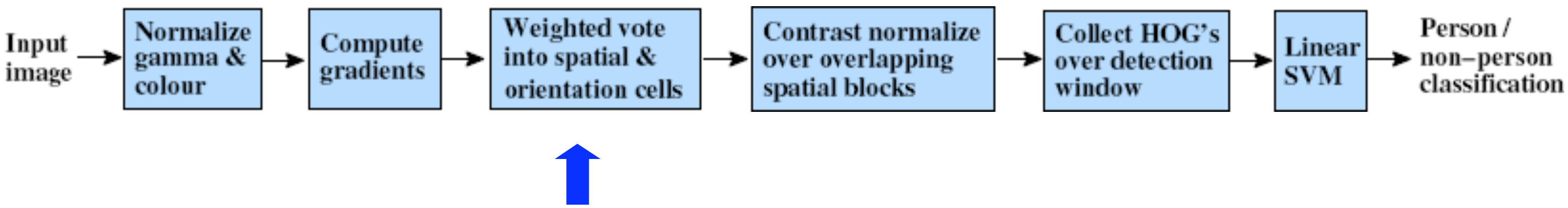
diagonal

-1	0	1
-2	0	2
-1	0	1

Sobel

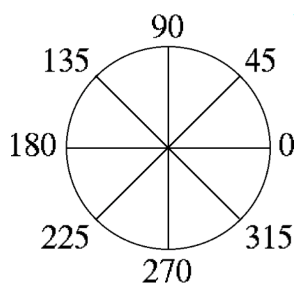
Slides by Pete Barnum

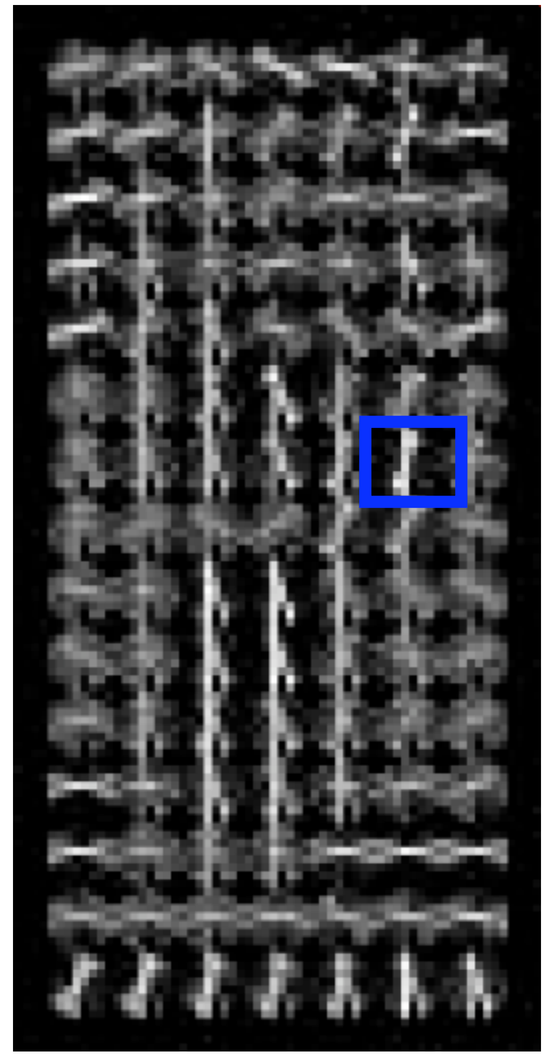
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



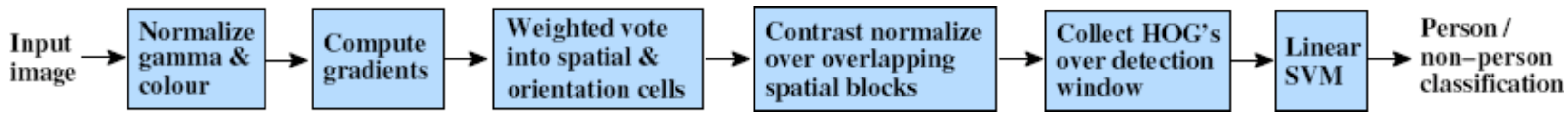
- Histogram of gradient orientations

  -Orientation



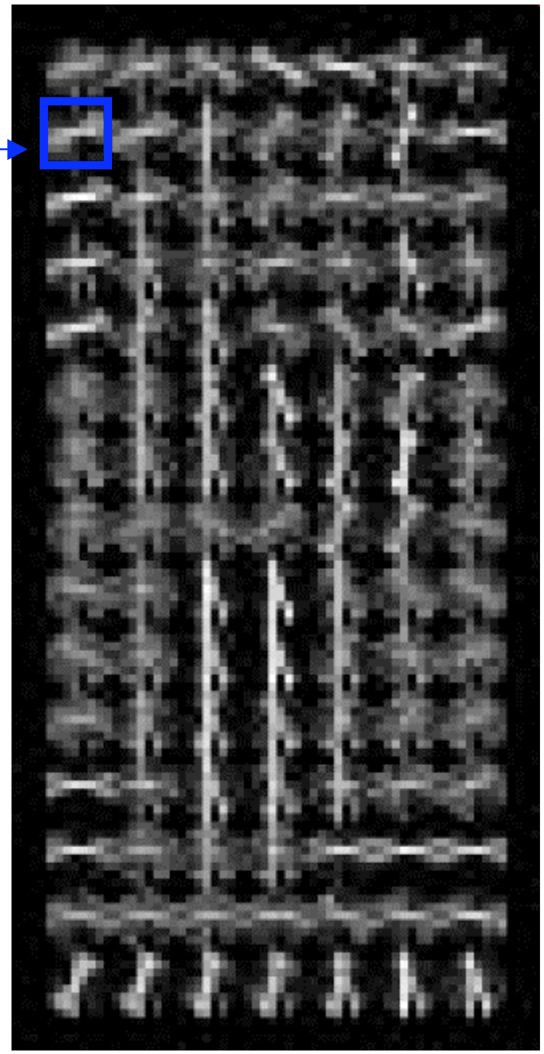


Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



8 orientations

X=

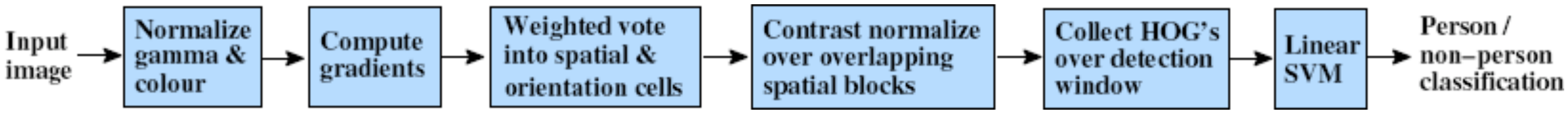


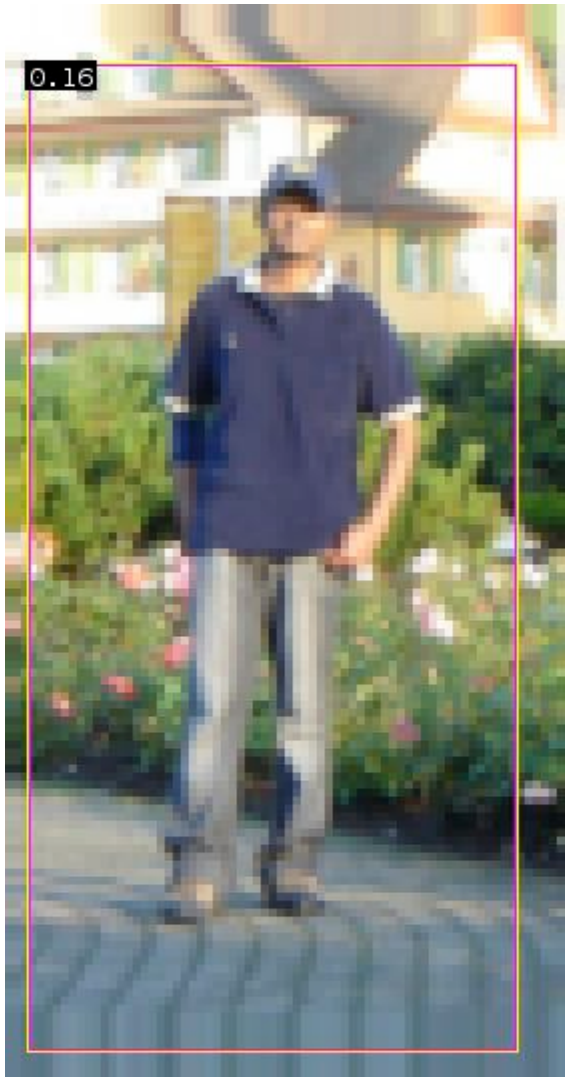
$\in R^{840}$

15x7 cells

Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05





$$0.16 = w^T x - b$$

$$sign(0.16) = 1$$

$=>$ pedestrian

Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

Kernel SVM

Decision function is a linear combination of support vectors:

$$w = \sum_{x_i - s.v.} \alpha_i x_i$$

Prediction is a dot product:

$$(w, x) = \sum_{x_i - s.v.} \alpha_i (x_i, x)$$

Kernel is a function that computes the dot product of data points in some unknown space:

$$(\Psi(x_i), \Psi(x)) = K(x_i, x)$$

We can compute the decision without knowing the space:

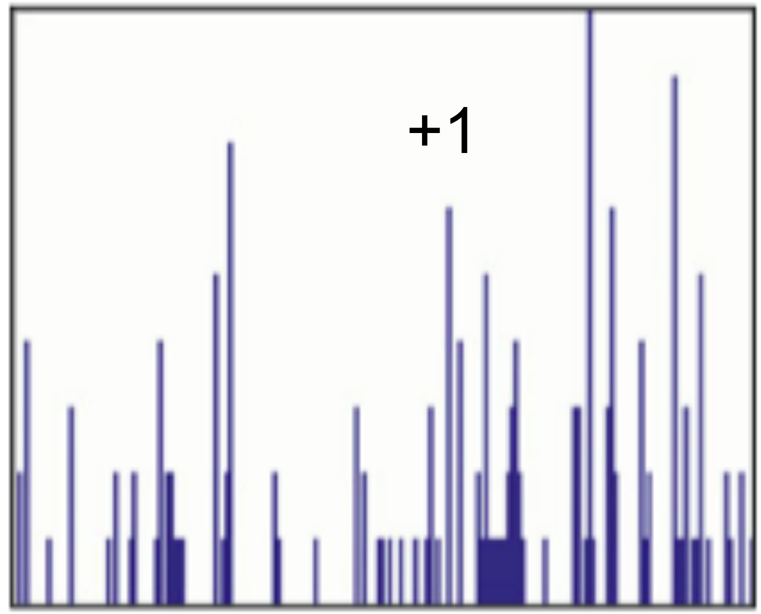
$$(w, \Psi(x)) = \sum_{x_i - s.v.} \alpha_i K(x_i, x)$$

Useful kernels

$$K(x_i, x) =$$

- Linear! (x_i, x)
- RBF $exp(-||x_i - x||^2/2\sigma^2)$
- Histogram intersection $\sum_j min(x_i^{(j)}, x^{(j)})$
- Pyramid match

Histogram intersection

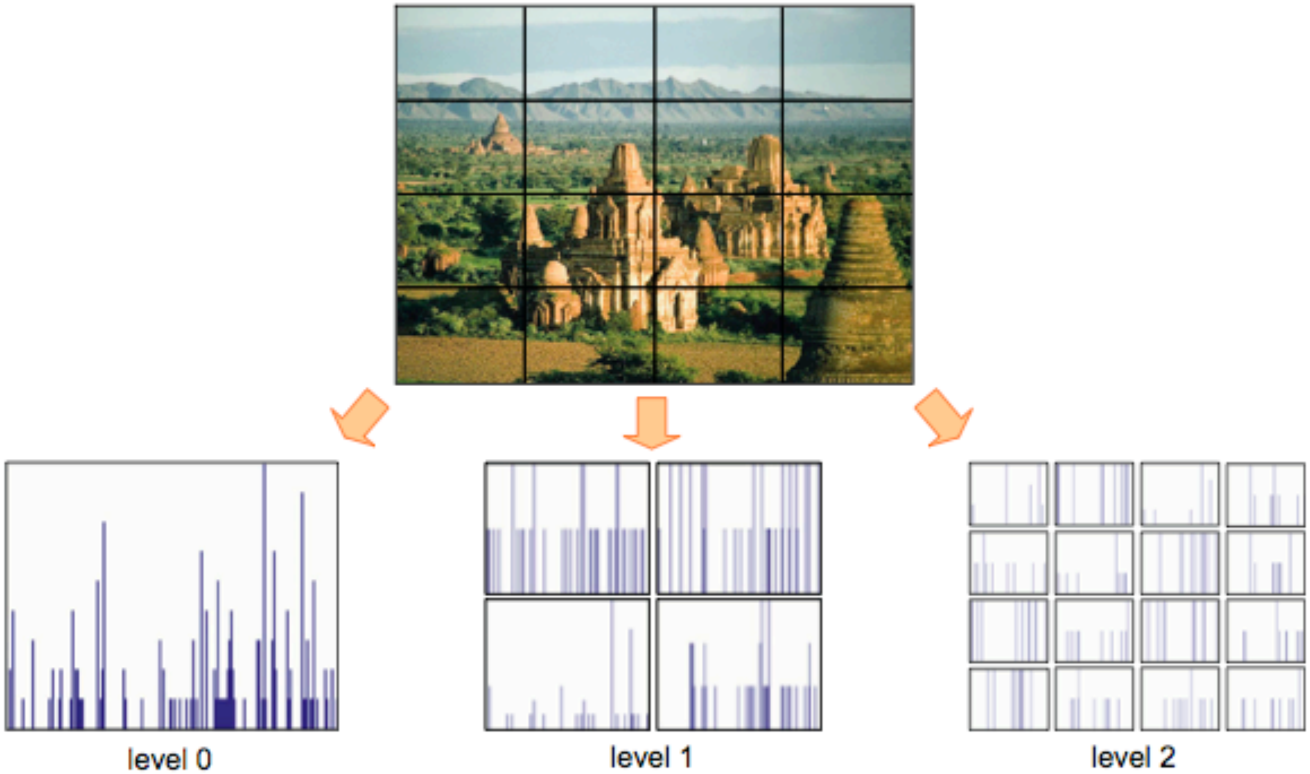


Assign to texture cluster → Count

$$HIK(x_i, x) = \sum_j min(x_i^{(j)}, x^{(j)})$$

S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories.

(Spatial) Pyramid Match



$$SPM(x_i, x) = \frac{1}{2^L} HIK_0(x_i, x) + ... + \frac{1}{L-l+1} HIK_l(x_i, x) + ... + HIK_L(x_i, x)$$

S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories.

Boosting

$$L(x, F(x)) = \{ \begin{array}{ll} 0, & y(x) = F(x) \\ 1, & otherwise \end{array}$$

$$Err(F, P(x)) = \int_{P(x)} P(x) L(x, F(x)) \approx \sum_{x_j} p_j L(x_j, F(x_j))$$

- Weak classifier

  Classifier that is slightly better than random guessing

$$Err(f, P(x)) \neq \frac{1}{2}$$

- Weak learner builds weak classifiers

$$\forall P(x) \quad \{y_i, P(x_i)\} \rightarrow WL \rightarrow f$$

Boosting

- Start with uniform distribution

$$(x_i, y_i),\ p_i^{k=1} = 1/N$$

- Iterate:
  1. Get a weak classifier f_k
  2. Compute it's 0-1 error
  3. Take
  4. Update distribution

$$\epsilon_k = \sum_{x_i} p_i L(x_i, f_k(x_i))$$

$$\alpha_k = \frac{1}{2}\ln(\frac{1-\epsilon_k}{\epsilon_k})$$

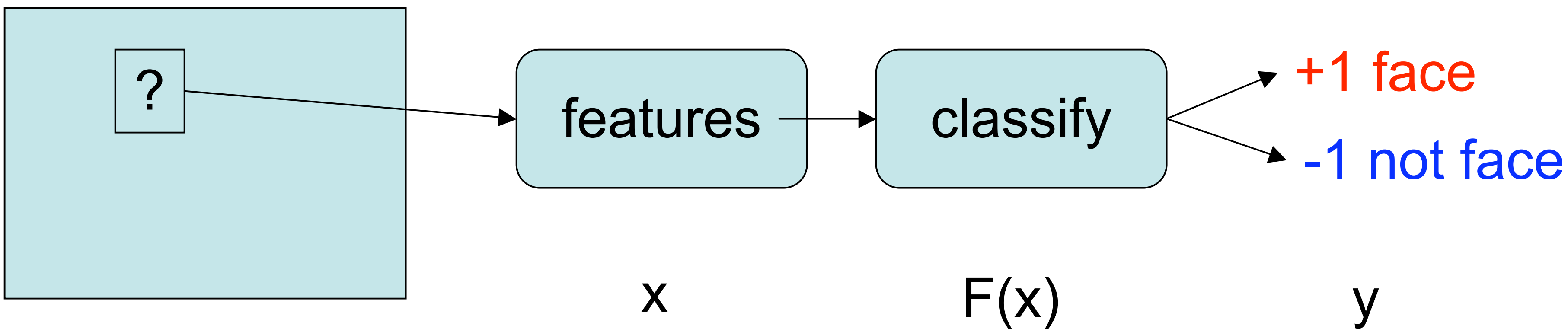
$$p_i^{k+1} = \frac{p_i^k exp(-\alpha_k y_i f_k(x_i))}{Z_{k+1}}$$

- Output the final "strong" classifier

$$F(x) = sign(\sum_k \alpha_k f_k(x))$$

Yoav Freund Robert E. Schapire, A Short Introduction to Boosting

Face detection



- We slide a window over the image
- Extract features for each window
- Classify each window into face/non-face

Face detection

- Use haar-like features
- Use decision stumps as week classifiers
- Use boosting to build a strong classifier
- Use sliding window to detect the face