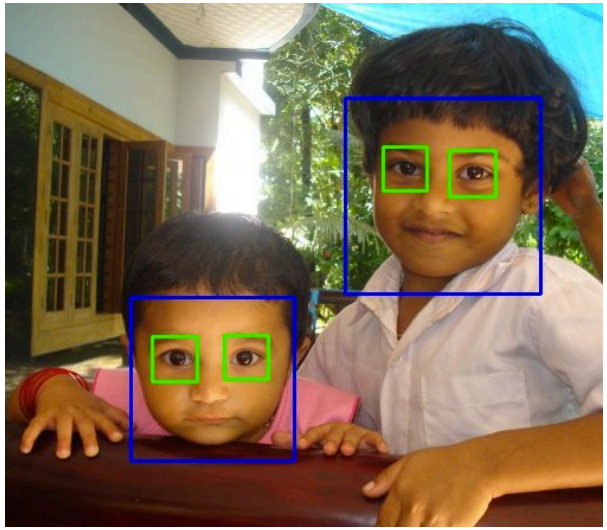
Cascading e Boosting

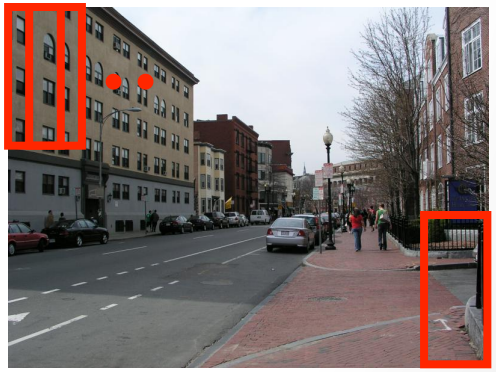
Prof. Dr. Geraldo Braz Junior

Objetivo

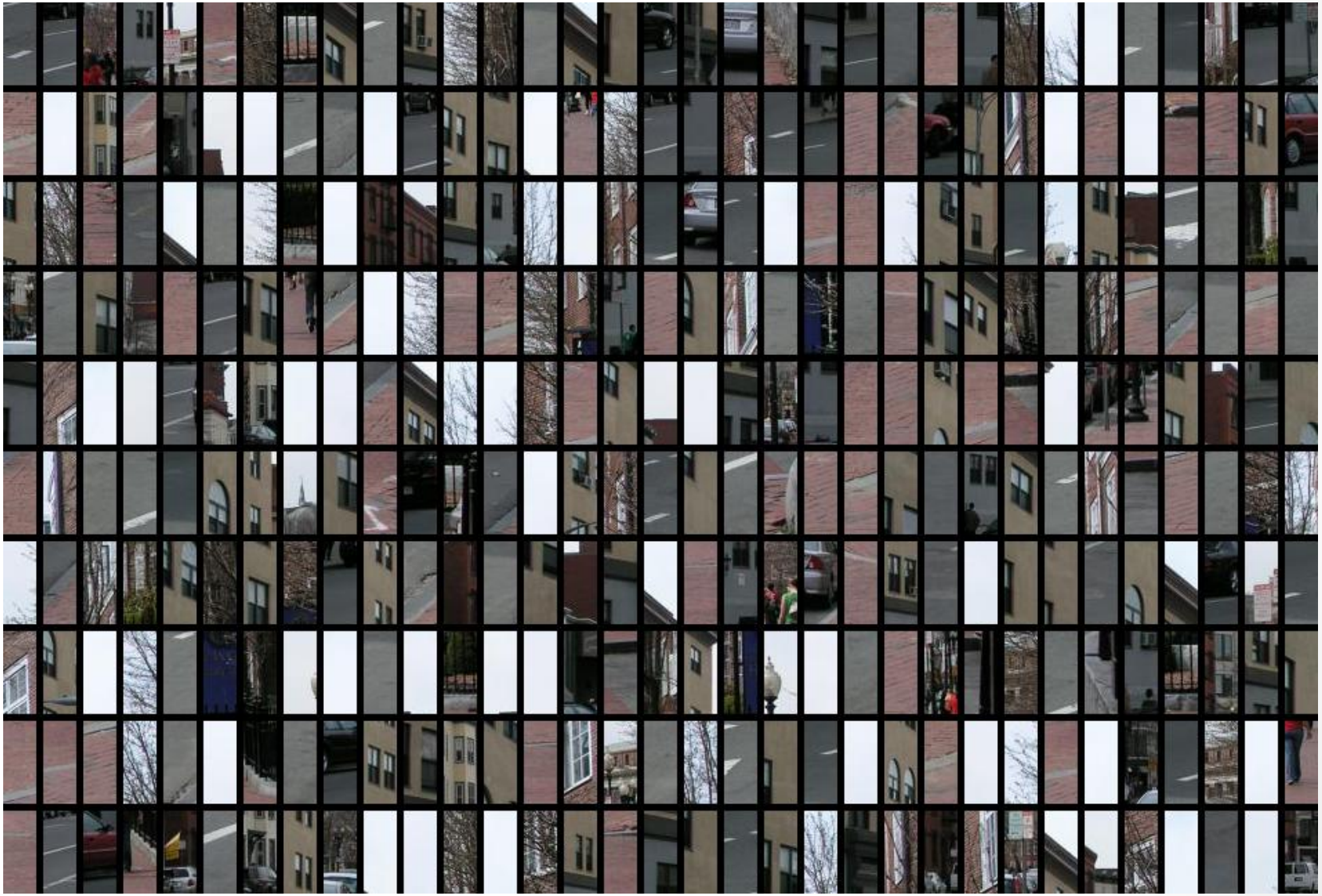
- Detectar instancias de objeto



Uma abordagem: janela deslizante



Quais são as janelas geradas?



Como reconhecer as janelas?

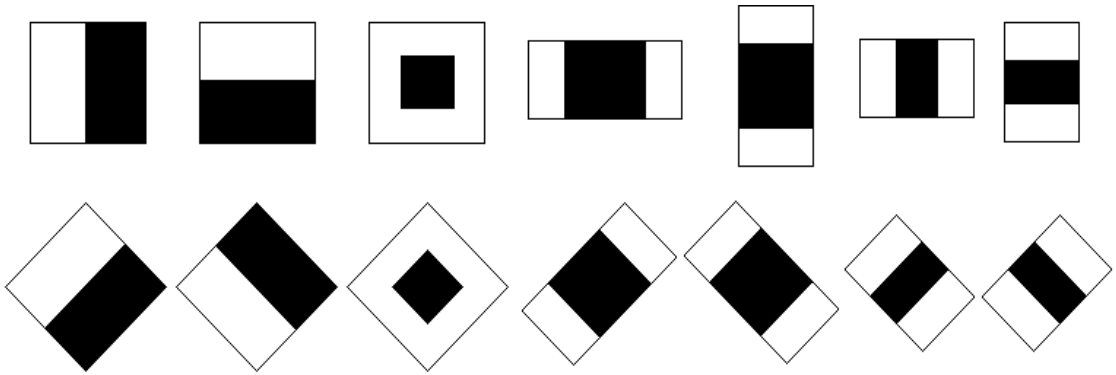
- Template Matching
- Shape
- **Cascade of Classifiers (+ Boosting)**
  - Proposto por Viola and Jones: "Rapid Object Detection using a Boosted Cascade of Simple Features" 2001

Característica básica

- Treino (lento), Teste (rápido)
  - Características simples: Haar (pode usar LBP, HOG, ou ….)
  - Gera 1 classificador fraco para cada característica que combinados tornam-se fortes: Boosting
  - Cria vários classificadores fortes em sequência: cascading

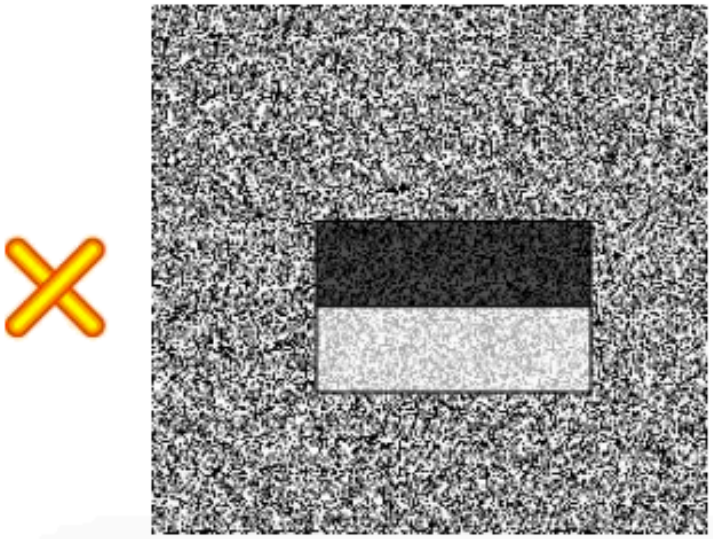
Haar Features

- Diferença da soma de “brancos” com “pretos”
- O filtro deve ser posicionado na imagem, numa escala específica
- E depois obtido a métrica



Haar Features

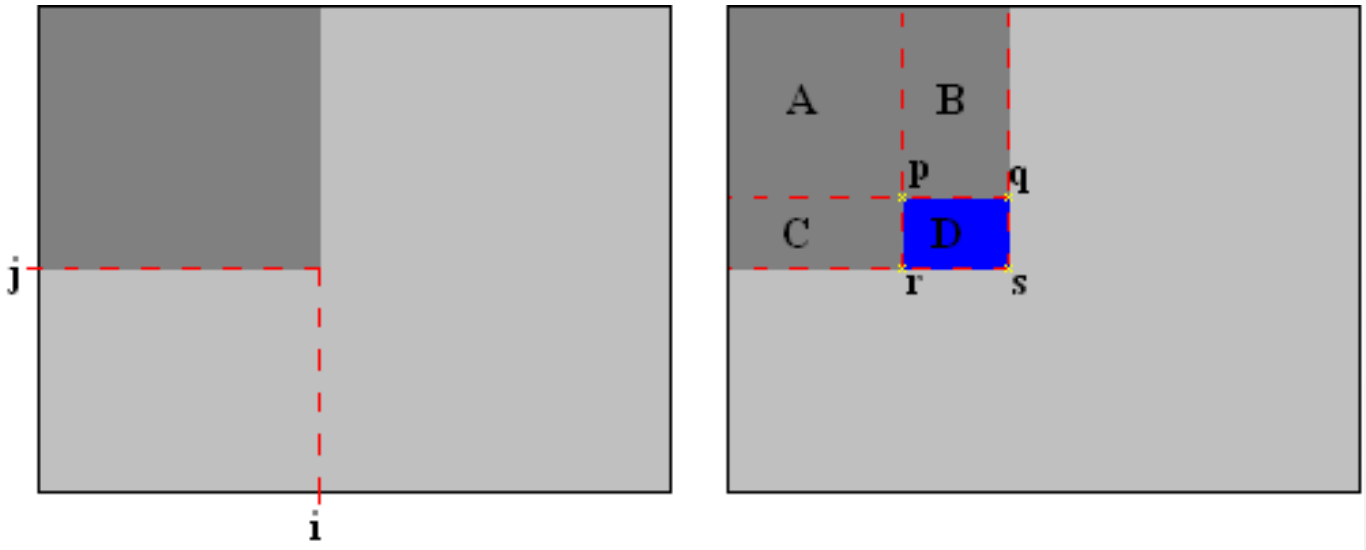
- Melhores respostas em imagens com definição de formas, textura, …



Haar Features

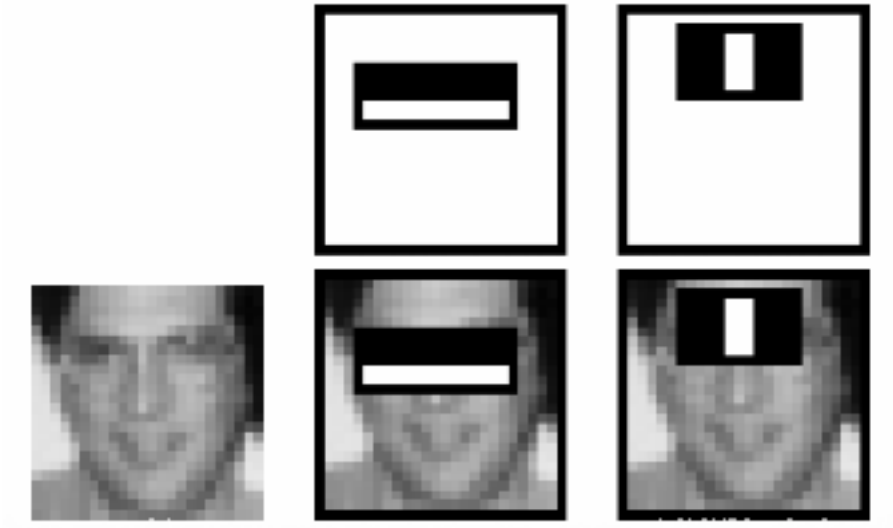
- Número excessivo de operações?
- Caso seja usado **imagem integral**, não.

$$sum(D) = ii(p) + ii(s) - ii(q) - ii(r)$$



Quantidade x Qualidade

- Se considerar todas as localizações e tamanhos de janela vai levar a geração de muitas features (~160000).
- Mas, quais são realmente boas?



Boosting para seleção

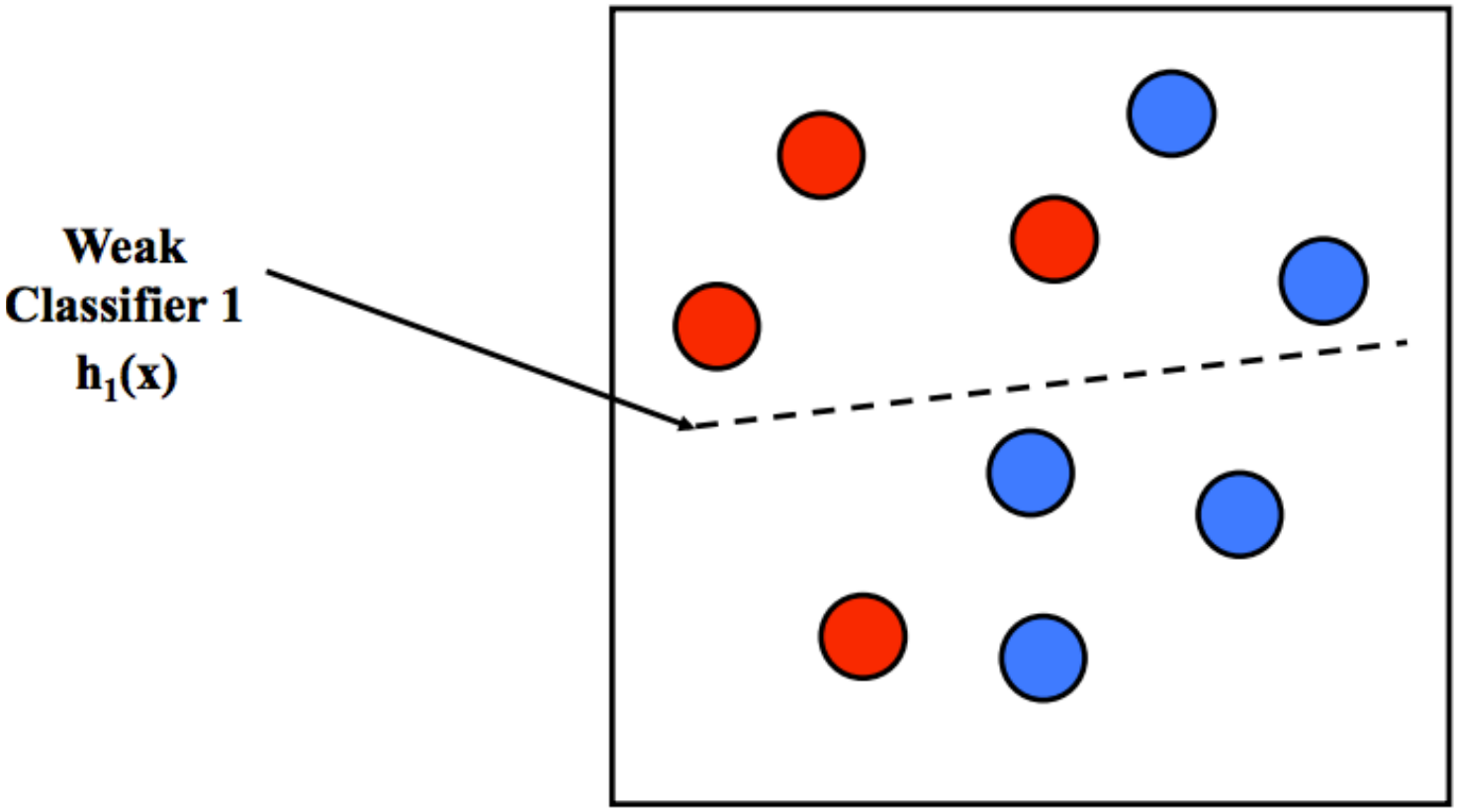
- Esquema de classificação (**ensemble**) que combina vários classificadores fracos para gerar um classificador robusto

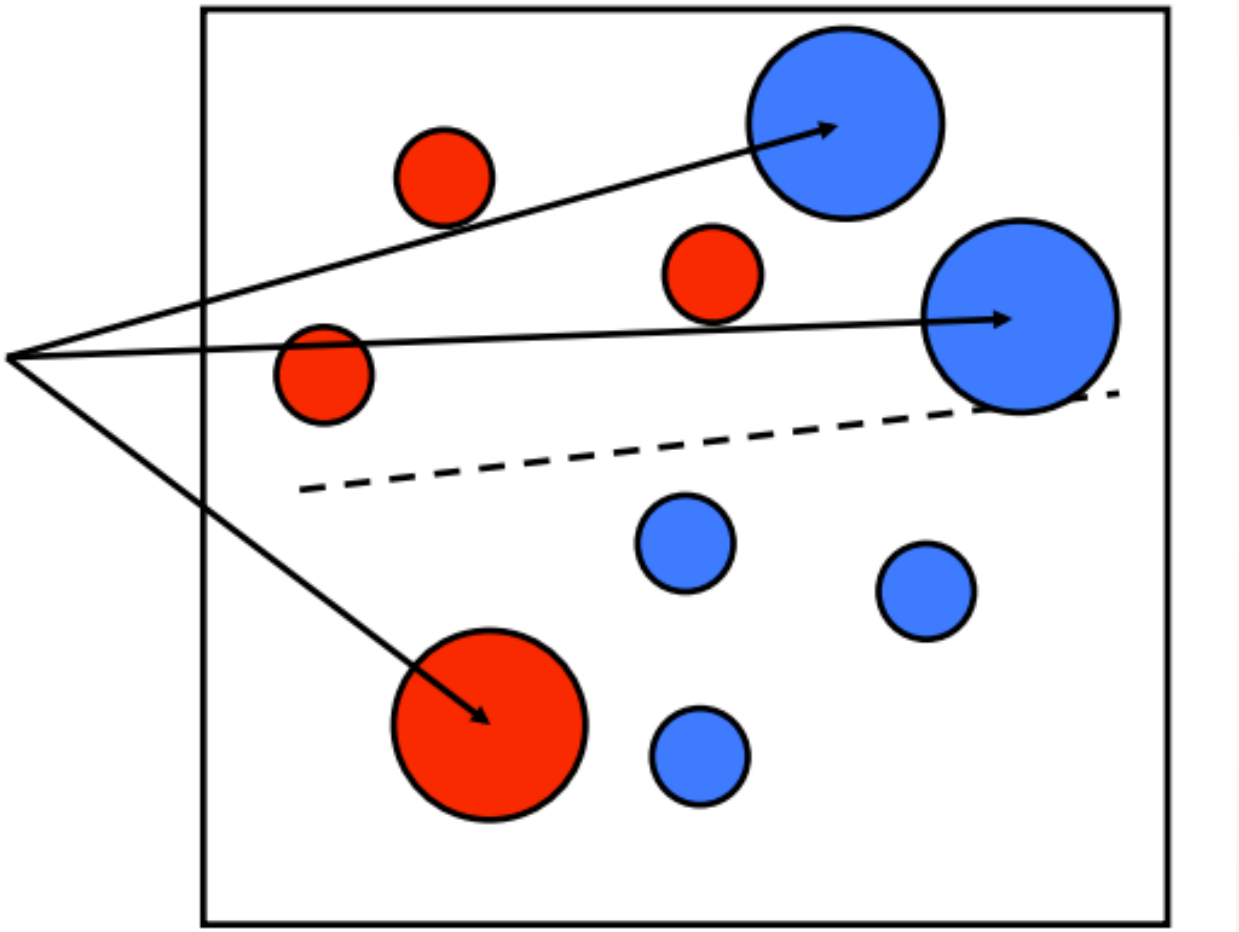
- Fraco = usar apenas 1 feature

- Boosting rounds
  - selecionar um novo classificador fraco, que tem resultado melhor do que seus antecessores

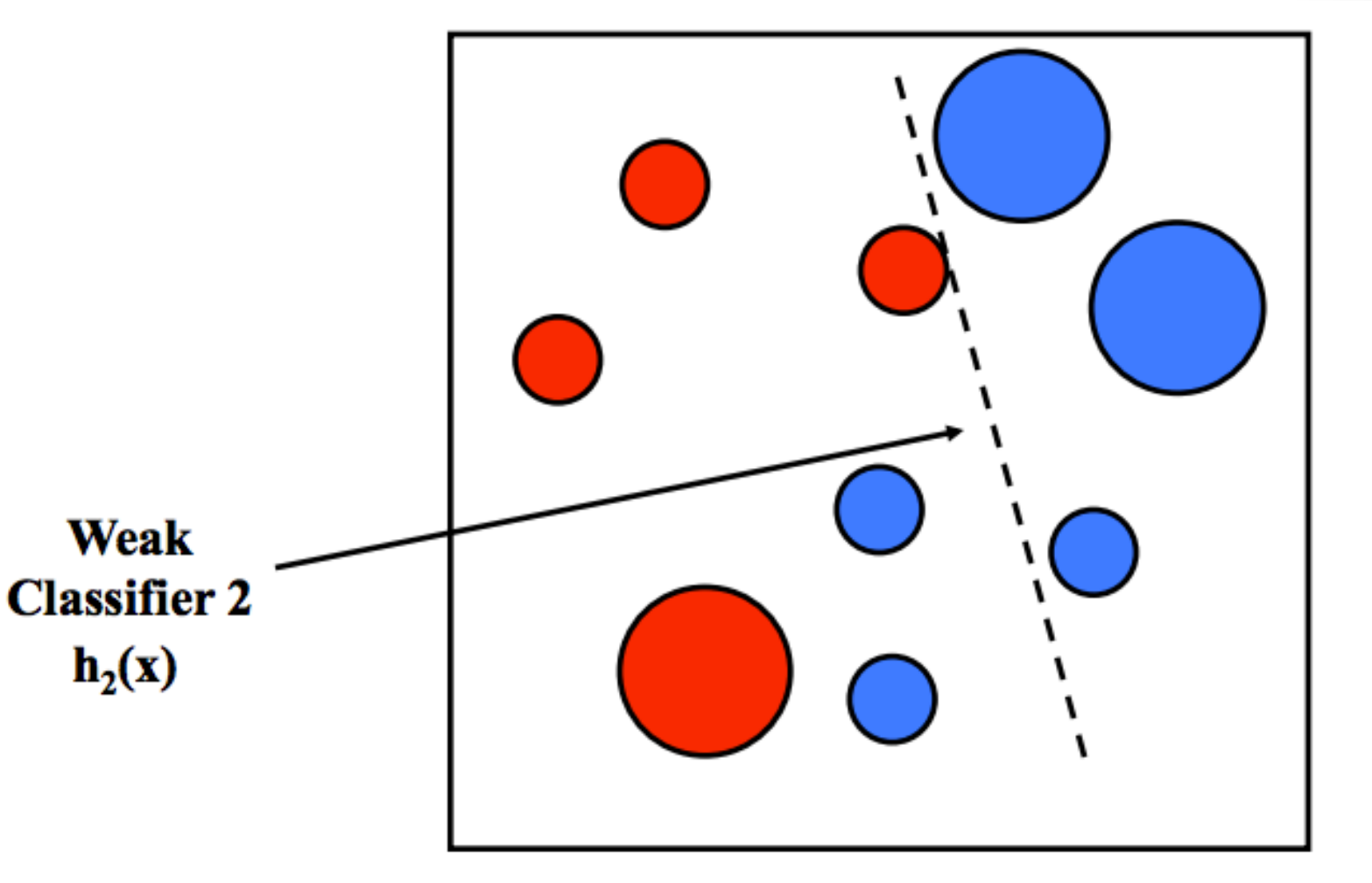
Boosting: o processo

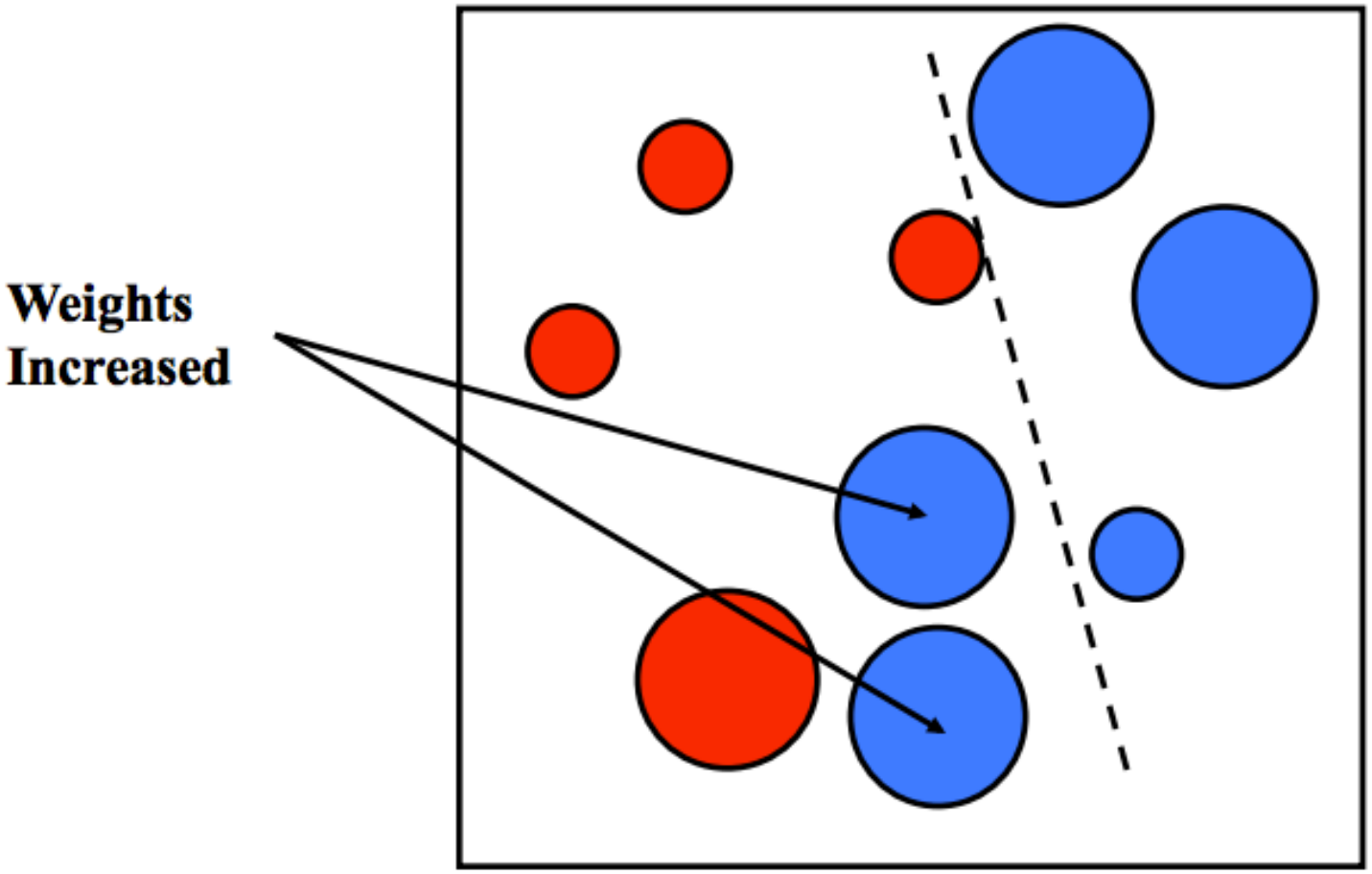
1. Inicializa todos os pesos de amostra igual
2. Interação
   a. procure pelo classificador fraco que minimize o erro
   b. incremente o peso de quem errou
   c. repita
3. O classificador final é a combinação linear (ponderada) dos classificadores fracos

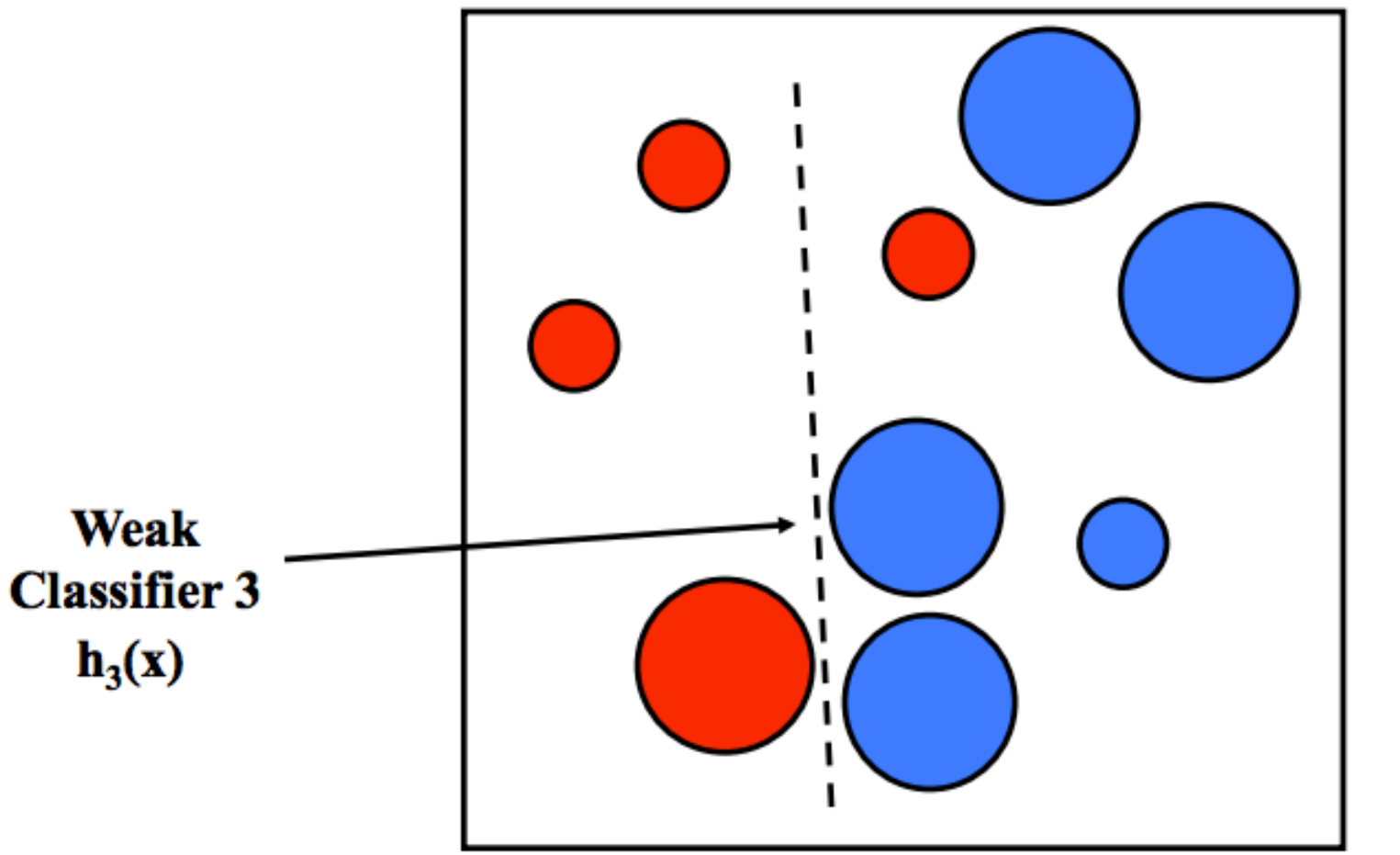

Weak
Classifier 1
$h_1(x)$



th

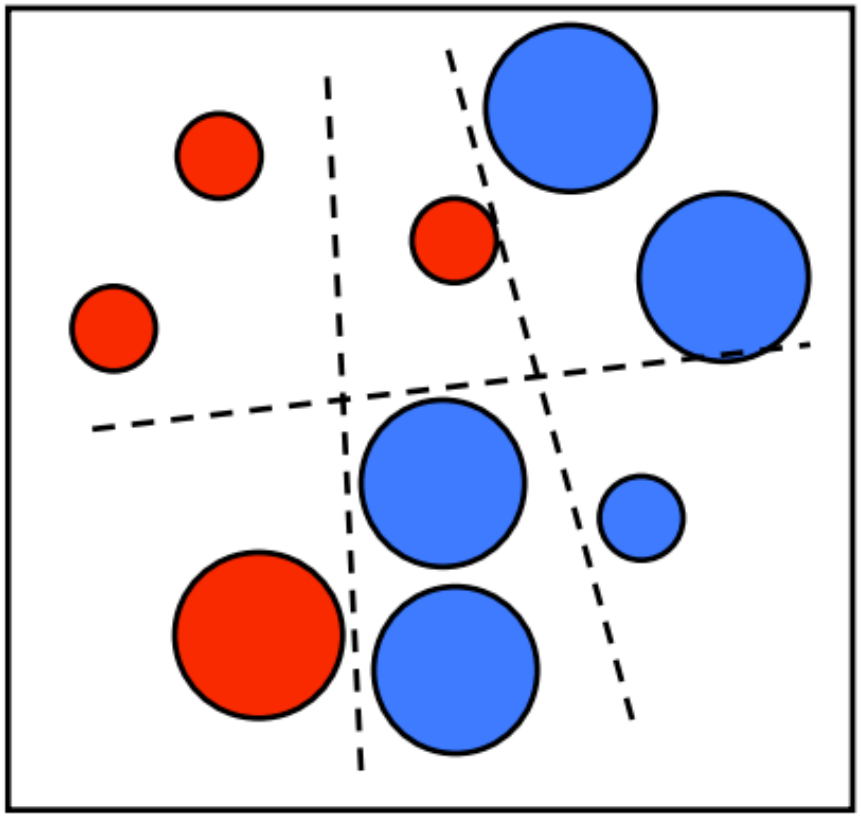

Weak
Classifier 2
$h_2(x)$


Weights
Increased


Weak
Classifier 3
$h_3(x)$

Final classifier is a weighted combination of the weak classifiers

$$(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{M} \alpha_j h_j(\mathbf{x})\right)$$

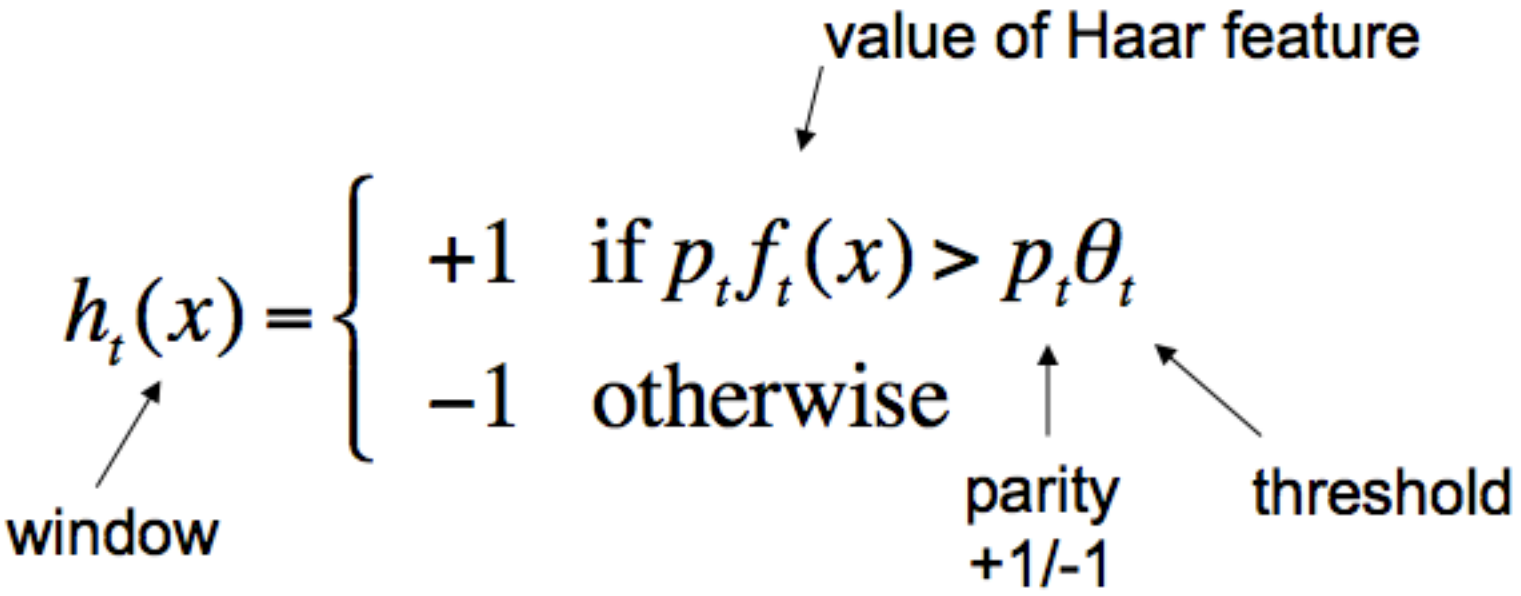


Boosting

- Vantagens
  - classificação com seleção de características
  - complexidade linear
  - teste rápido
  - fácil de implementar
- Problemas
  - precisa de **muitas amostras**
  - não funciona tão bem quanto outros classificadores, como SVM

Boosting aplicado a Detecção de Face

- ou problemas binários: estima o corte



Boosting aplicado a Detecção de Face

- Para cada rodada boost:
  - usa cada janela como exemplo
  - seleciona o melhor threshold para cada feature
  - seleciona a melhor combinação feature/ threshold como classificador fraco
  - calcula os pesos das amostras

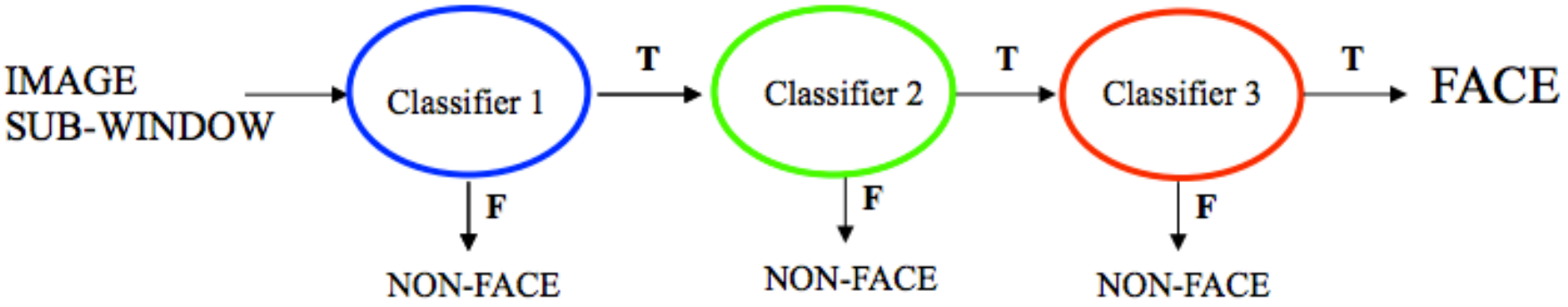
A maldição dos falso positivos

- Um classificador com as 2 melhores features pode ter 100% de detecção, mas com 50% de falso positivos (0.5 a cada imagem)

- Um classificador com as 200 melhores features pode obter 95% de detecção com 1 falso positivo a cada 14084

Cascade of Boosted

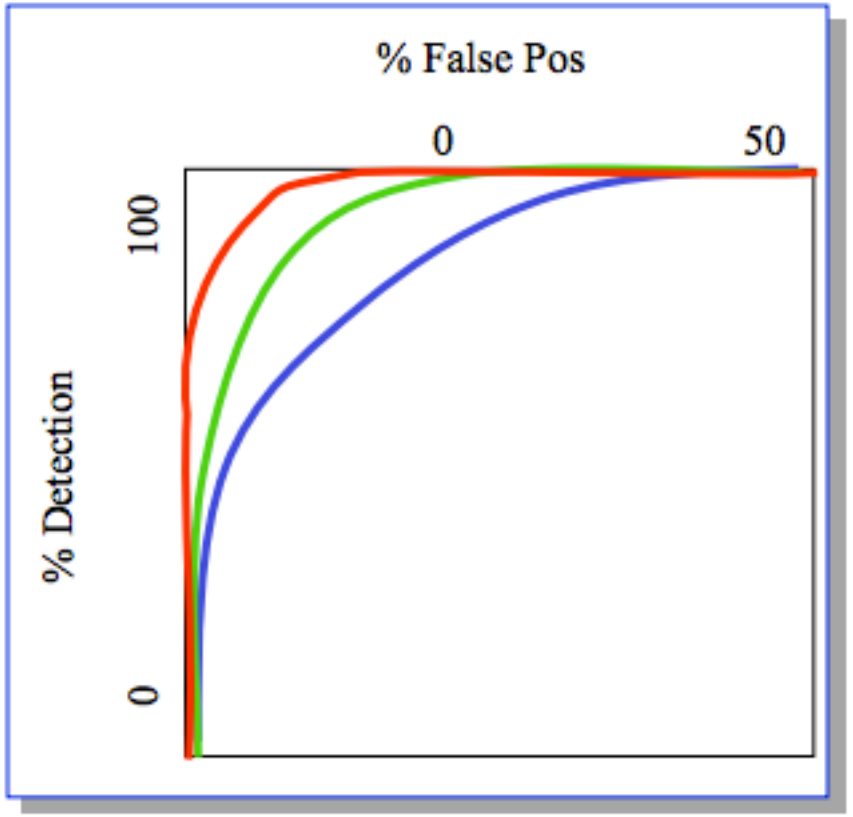
- Sequência de classificadores boosted com aumento constante de complexidade



- Negativos são eliminados imediatamente!

Cascade of Boosted

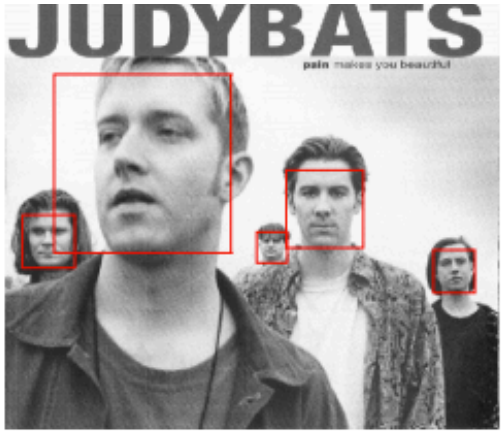
- Porque?
  - Melhoram o desempenho
  - Eliminam maior quantidade de falso positivos

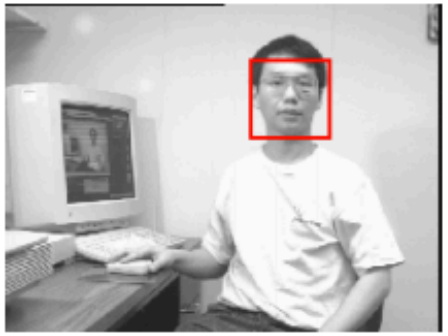


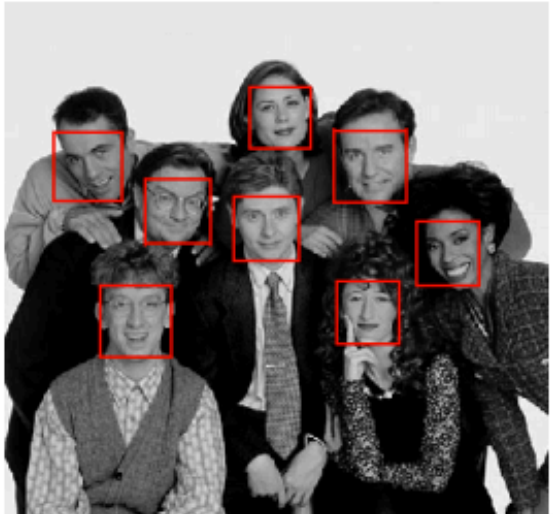
O processo de treinamento Cascading

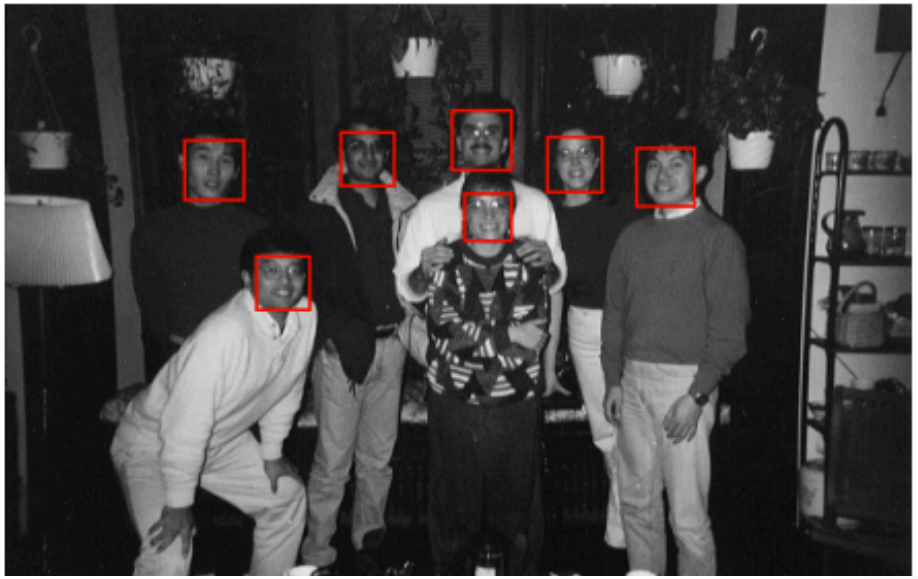
1. Informe a quantidade de estágios e a quantidade aceitável de FP para cada um
2. Adicione features para um estado até que ele chegue ao FP desejado
   - O teste é sempre num conjunto de validação
3. Se não conseguir (2), adicione um novo estágio
4. Use falso positivos de um estado como conjunto negativo do estágio futuro

Alguns resultados de Viola and Jones









Os testes deles

- Todas as ROIS são normalizadas
  - 5000 faces, 24x24, todas frontais
  - 9500 não faces

No OpenCV

- Para perfis já treinados no Opencv:
- Objeto:
  - **CascadeClassifier**
- Função:
  - **detectMultiScale**
- Exemplo: http://docs.opencv.org/3.0-beta/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html

Para criar seu próprio detector

1. Obtenha a base positiva e negativa
   a. a base positiva deve possuir a **localização de cada caso**
   b. a base negativa deve incluir também exemplos de fundo
2. Use **opencv_createsamples** para criar a base positiva
   a. Uma instancia por vez!!!!!
3. Use **opencv_traincascade** para fazer o treinamento
   a. pode especificar Haar, LBP ou HOG

DOC: http://docs.opencv.org/3.0-beta/doc/user_guide/ug_traincascade.html

Mais

- Variante muito boa que existe no opencv:

- **LatentSVM**: Discriminatively Trained Part Based Models for Object Detection
  - http://www.cs.berkeley.edu/~rbg/latent/index.html
  - http://docs.opencv.org/2.4/modules/objdetect/doc/latent_svm.html