Mecanismos de Detecção de Objetos – Selective Search

Visão Computacional

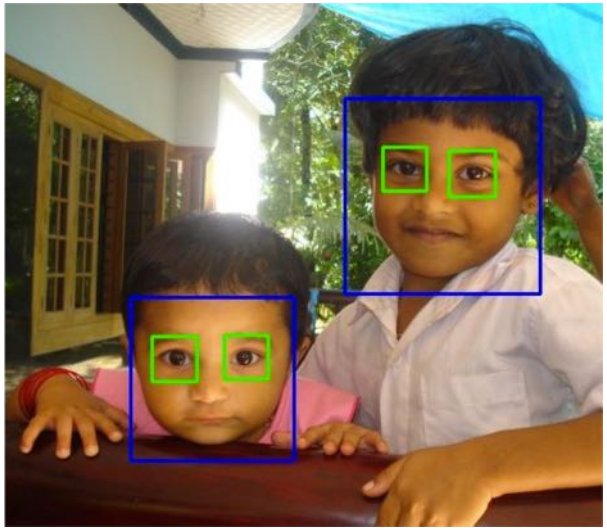
Programa de Pós-Graduação em Ciência da Computação – UFMA

Prof. Geraldo Braz Junior

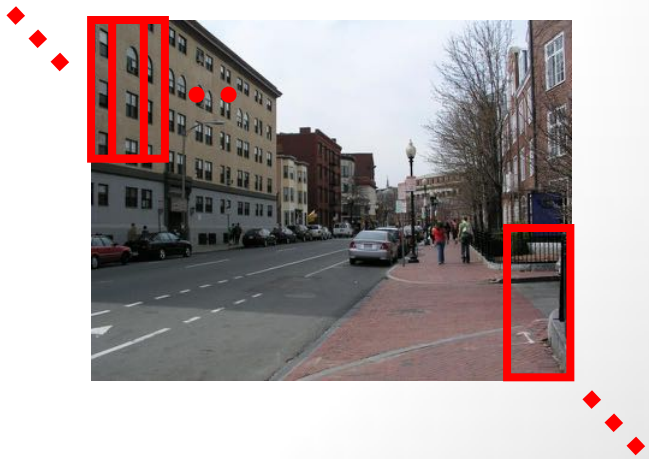
Contém material das notas de aula do CS131, CS229 CS231B

Objetivo

- Como detectar instancias de objeto?



Uma abordagem: janela deslizante



Quais são as janelas geradas?

Como reconhecer as janelas?

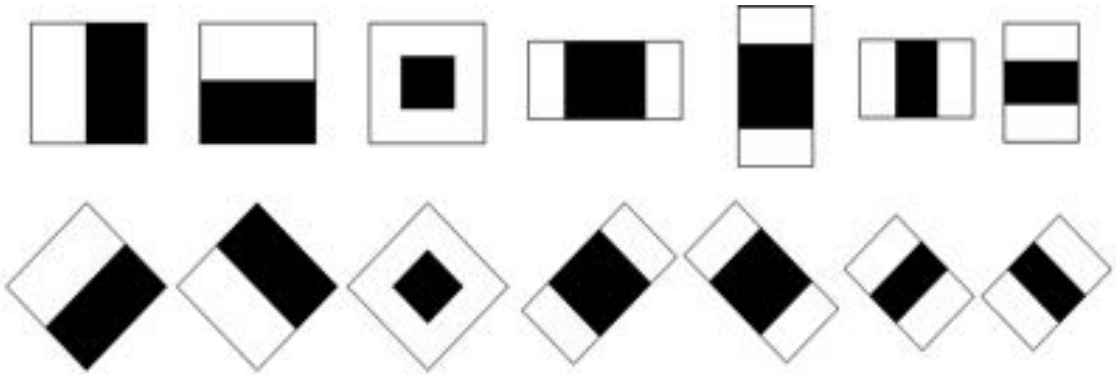
- Template Matching
- Shape
- **Cascade of Classifiers (+ Boosting)**
  - Proposto por Viola and Jones: "Rapid Object Detection using a Boosted Cascade of Simple Features" 2001

Característica básica

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

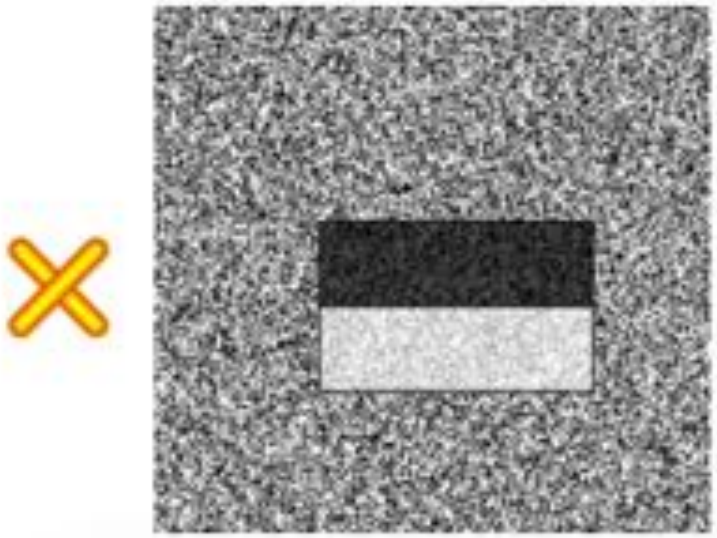
Haar Features

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



Haar Features

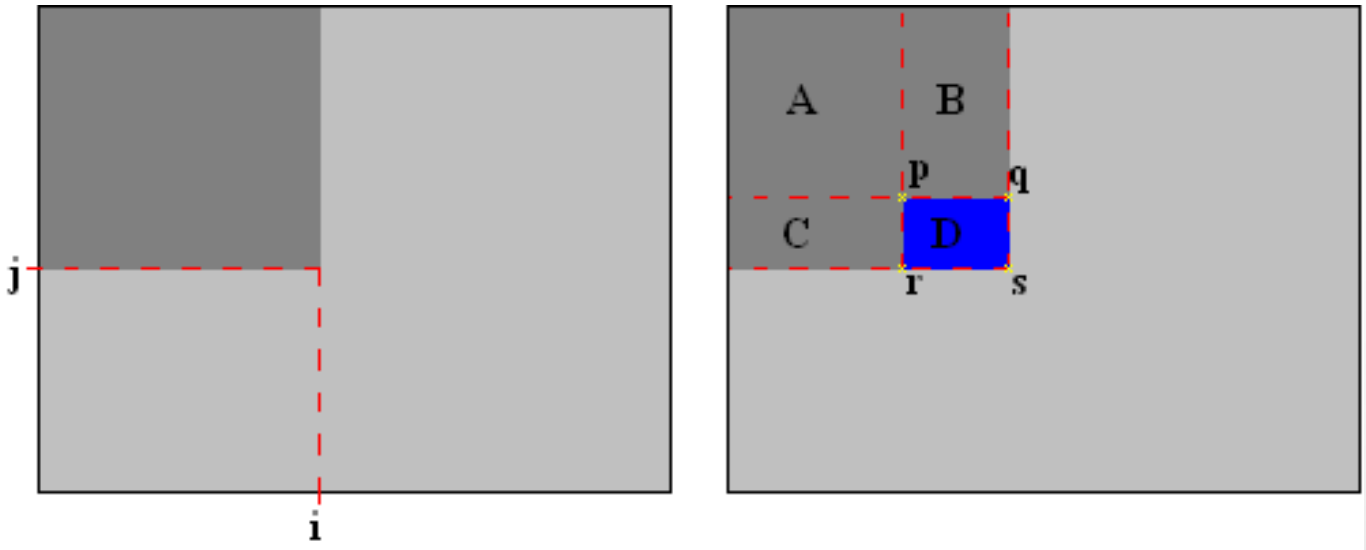
- Melhores respostas em imagens com definição de formas, textura, …



Haar Features

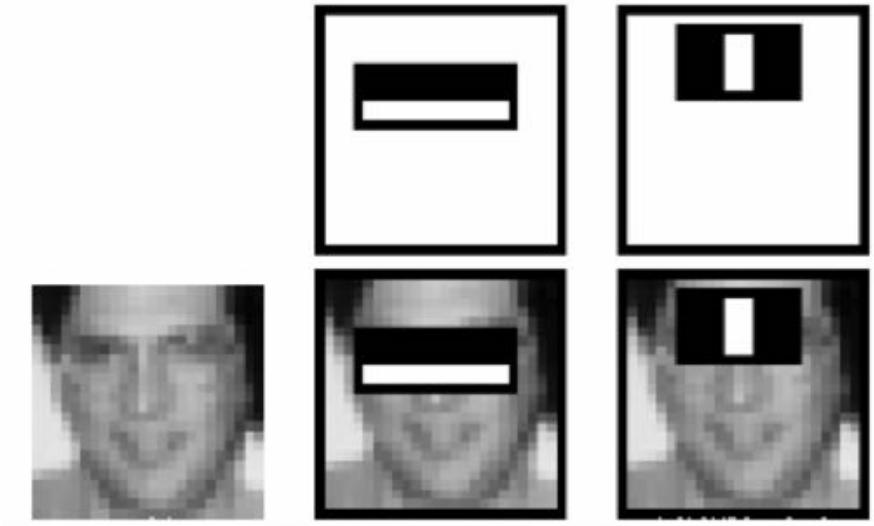
- Número excessivo de operações?
- Caso seja usado **imagem integral**, não.

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



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

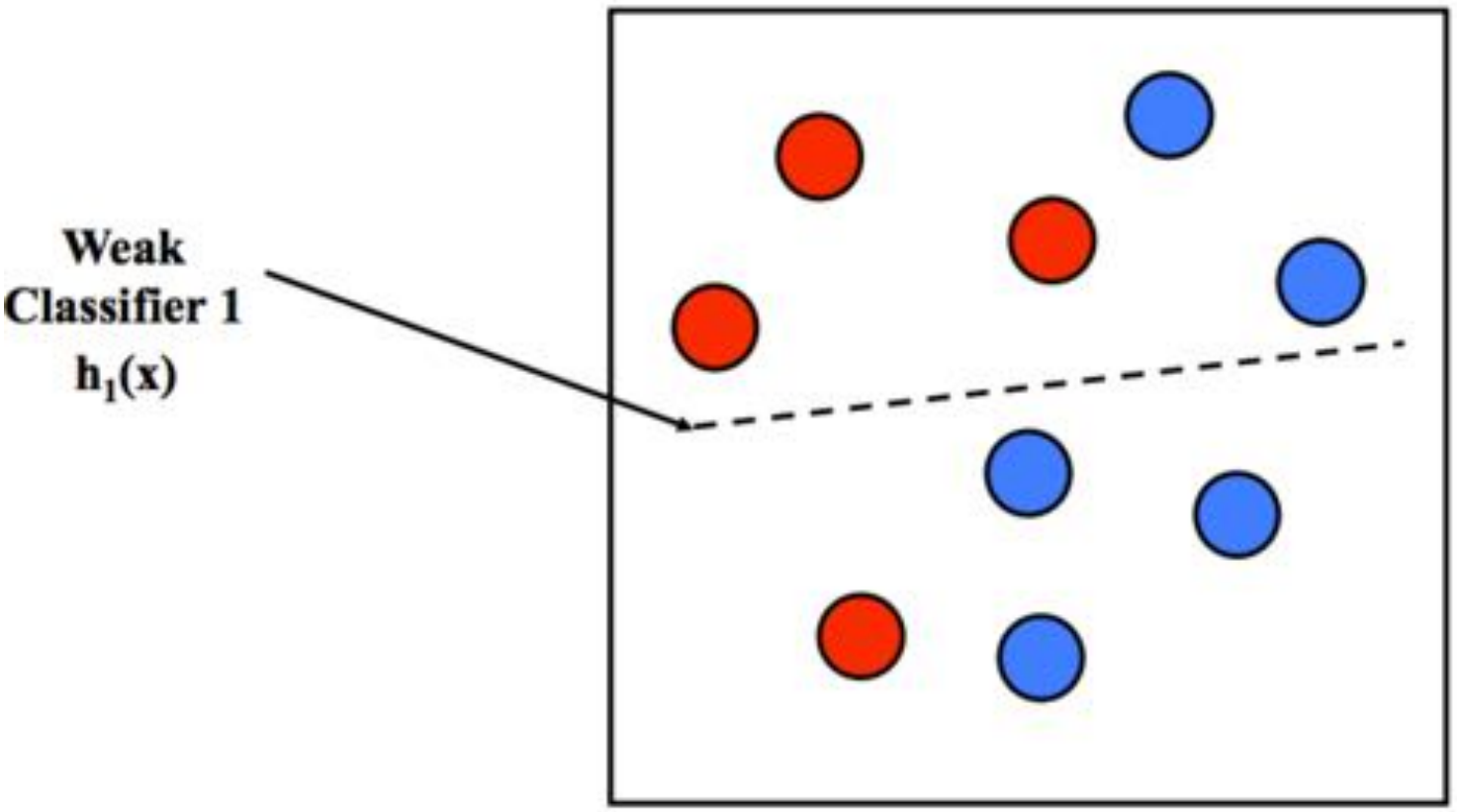
- Esquema de classificação (**ensemble**) que combina vários classificadores fracos para gerar um classificador robusto

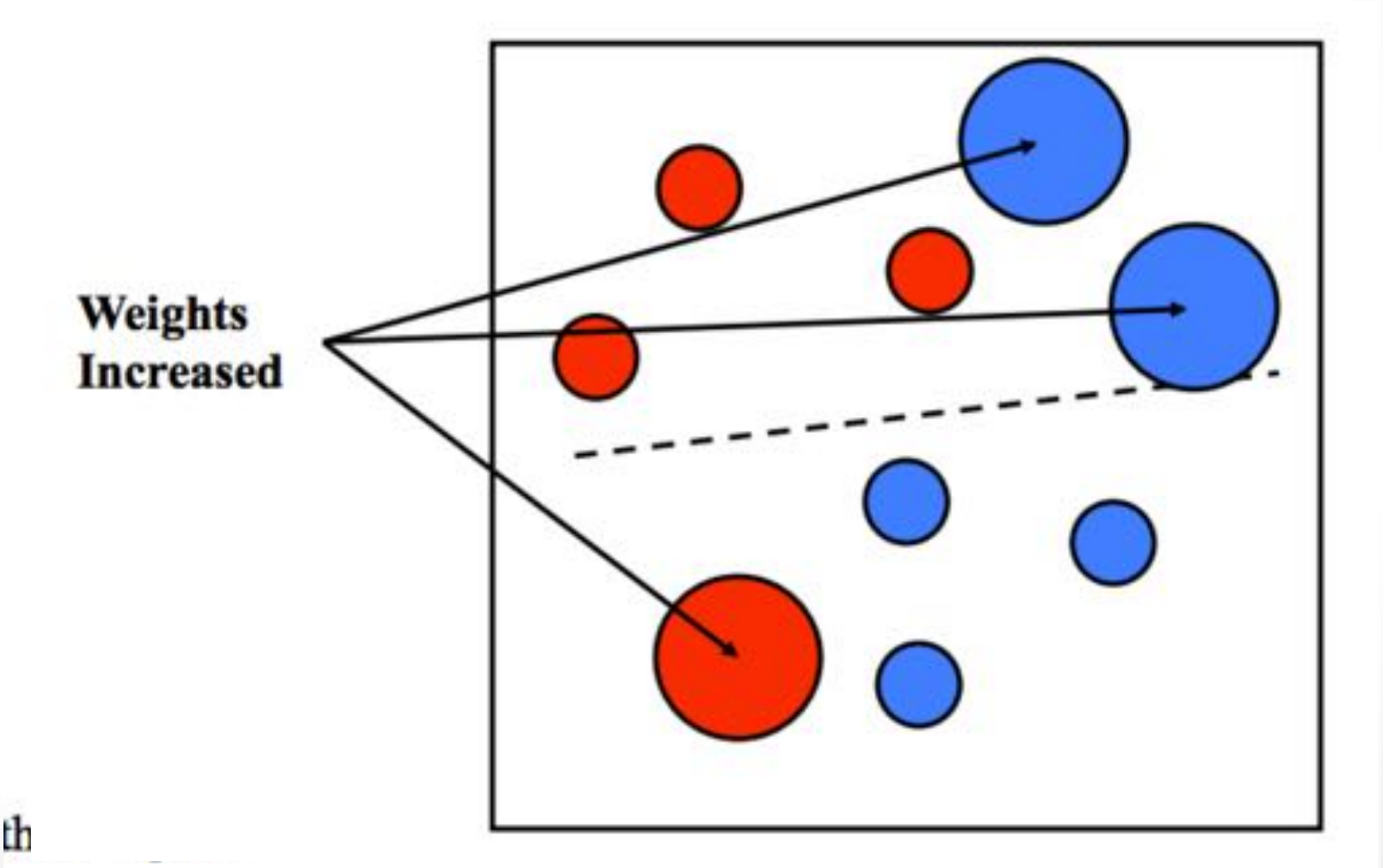
- Fraco = usar apenas 1 feature

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

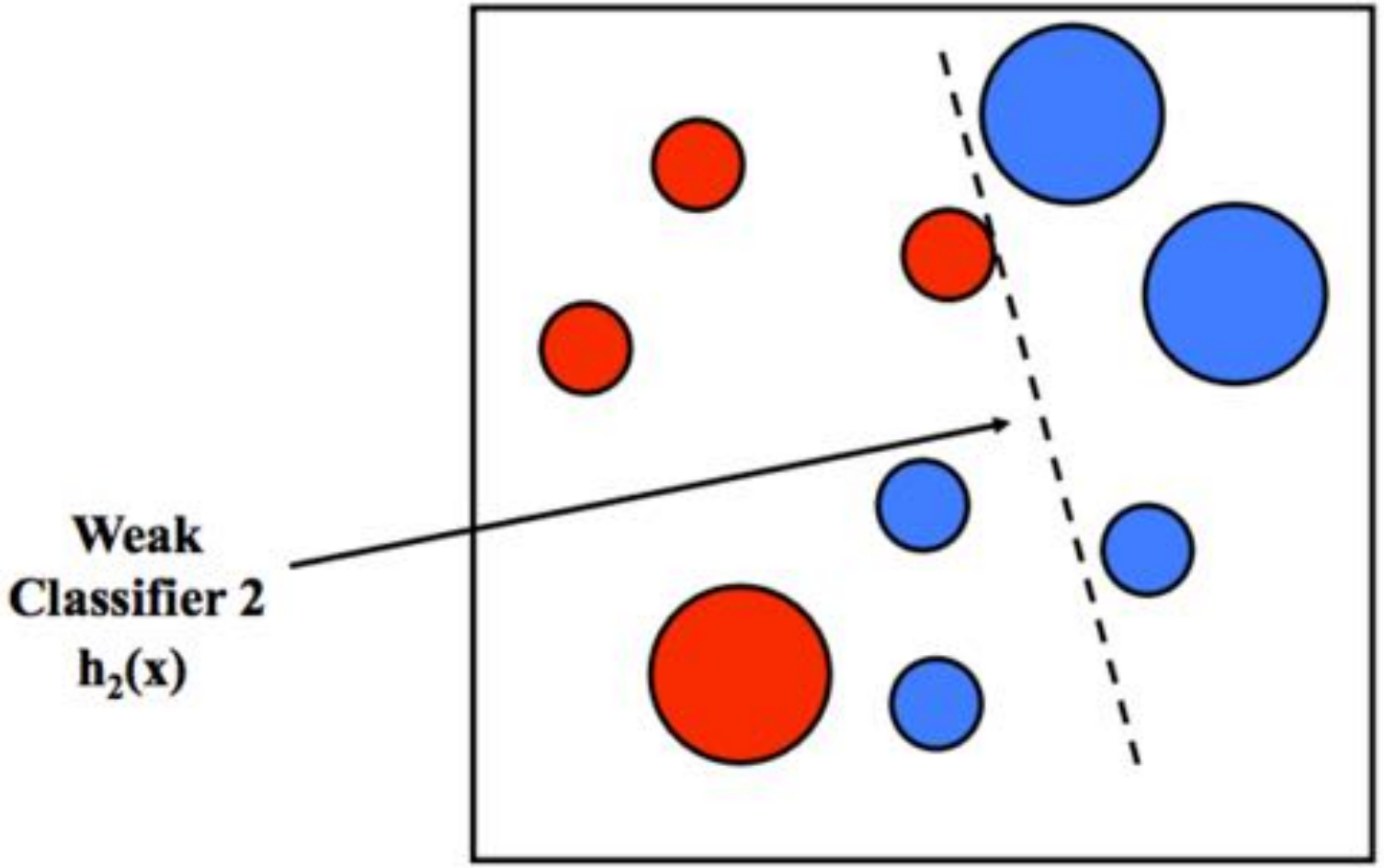
Boosting: o processo

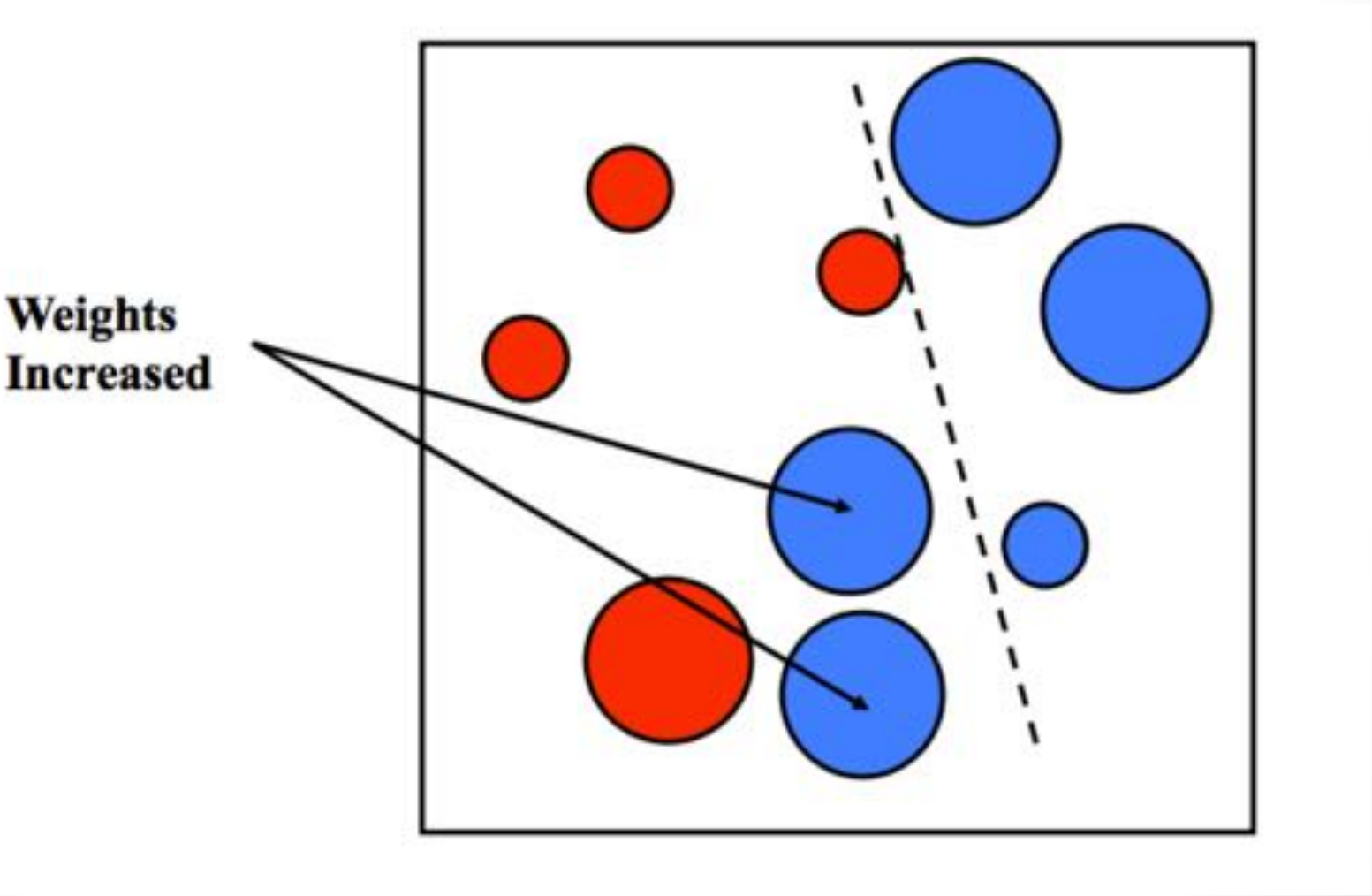
1. Inicializa todos os pesos de amostra igual
2. Interação
   a. procure pelo classificador fraco que minimize o erro
   b. incremente o peso de quem errou
   c. repita
3. O classificador final é a combinação linear (ponderada) dos classificadores fracos

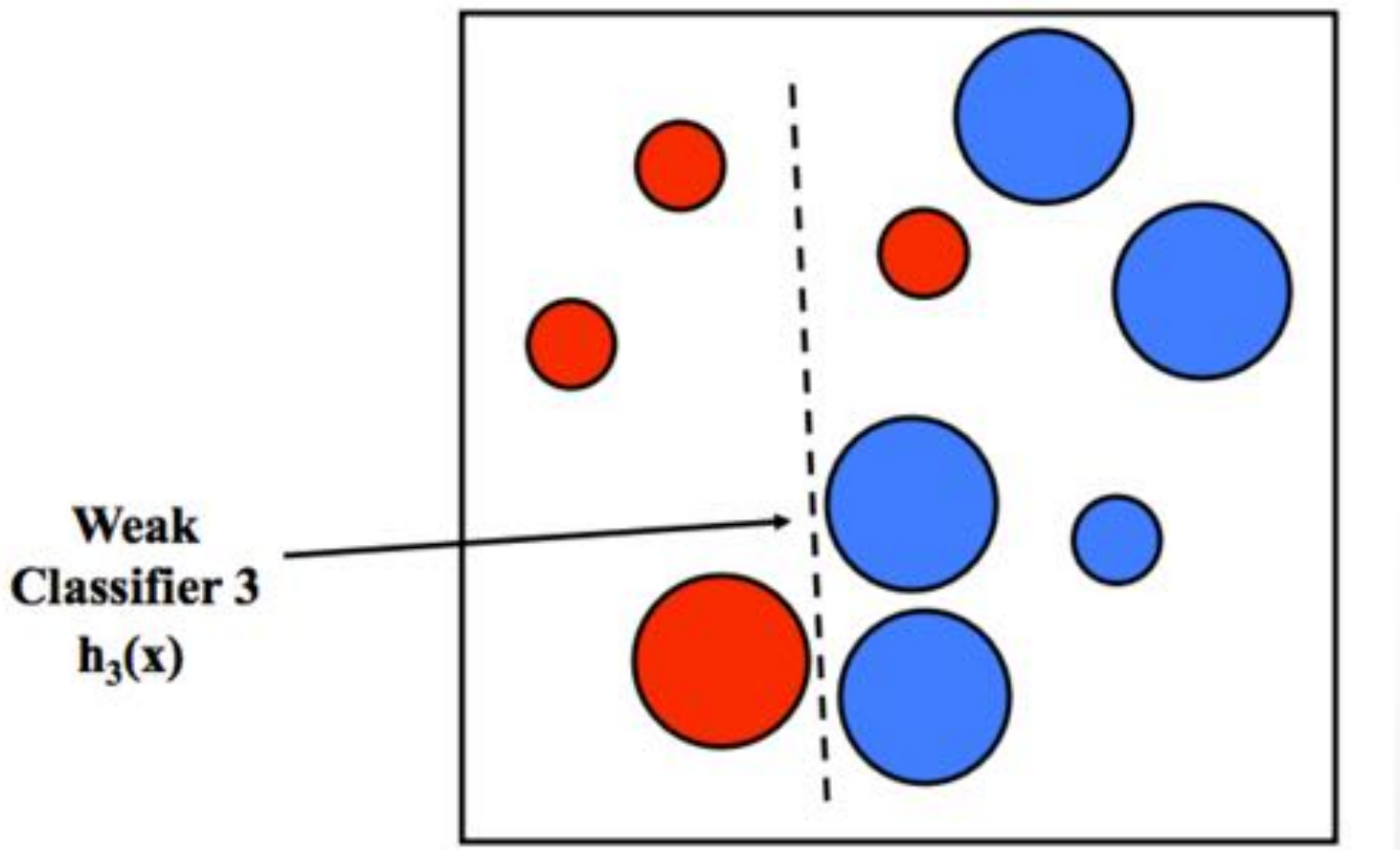

Weak
Classifier 1
$h_1(x)$



th

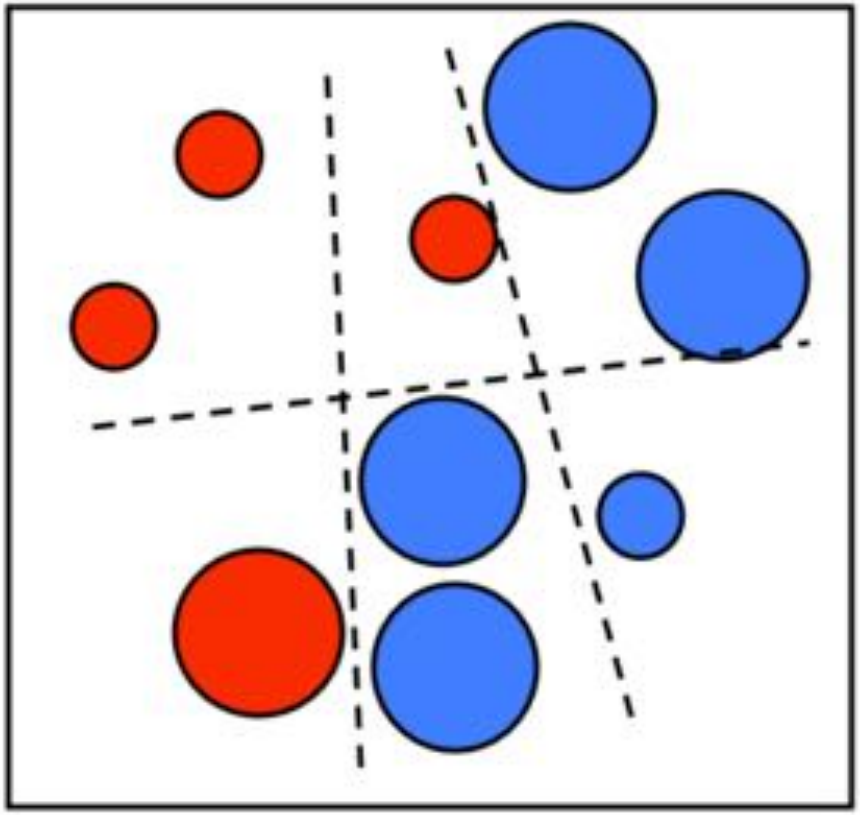

Weak
Classifier 2
$h_2(x)$


Weights
Increased


Weak
Classifier 3
$h_3(x)$

Final classifier is a weighted combination of the weak classifiers

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

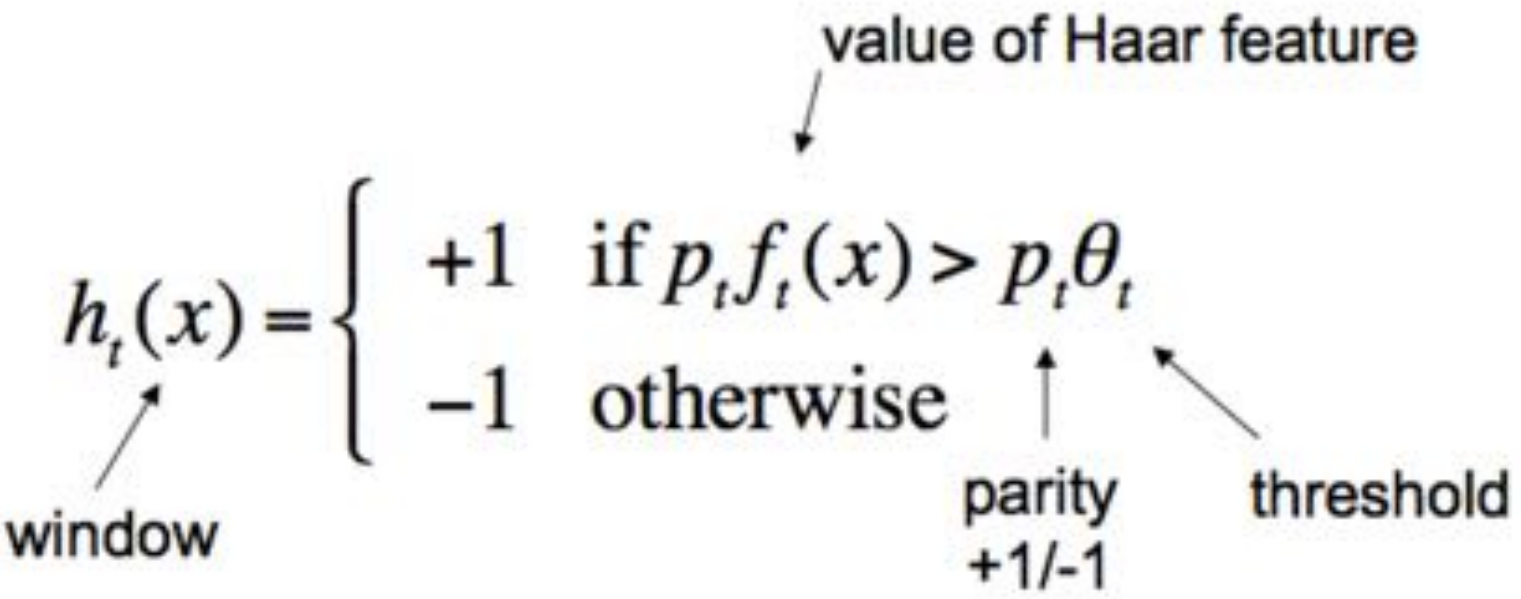


Boosting

- Vantagens
  - classificação com seleção de características
  - complexidade linear
  - teste rápido
  - fácil de implementar
- Problemas
  - precisa de **muitas amostras**

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

Na prática

- Ache o melhor classificador $h_t(x)$ que tenha o menor erro global ponderado (usando alguma função de erro)
- Calcule alpha $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- Adicione a nova função no ensemble
  - $F_t(x) = F_{t-1}(x) + \text{alpha} * h_t(x)$
- Atualize os pesos (e depois normalize os pesos 0-1!)

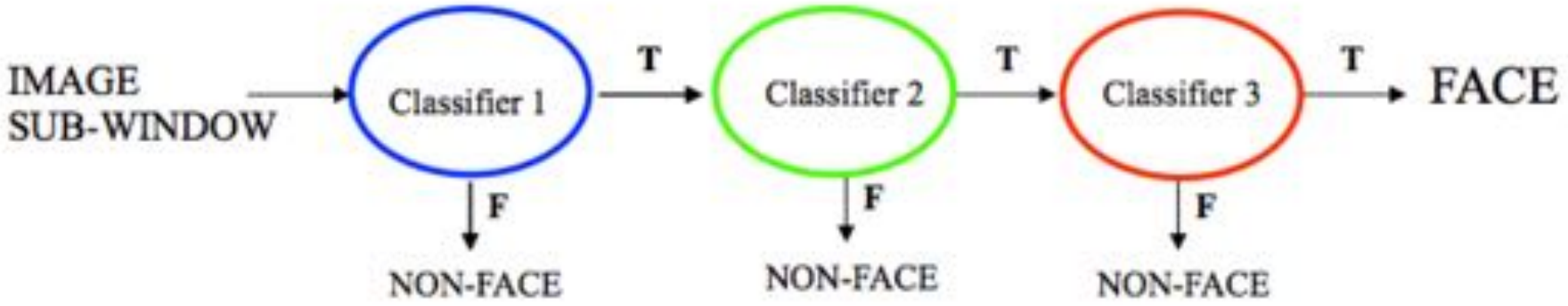
A maldição dos falso positivos

- 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

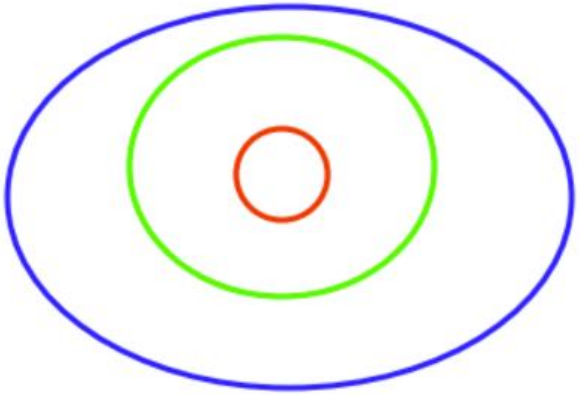
- Sequência de classificadores boosted com aumento constante de complexidade

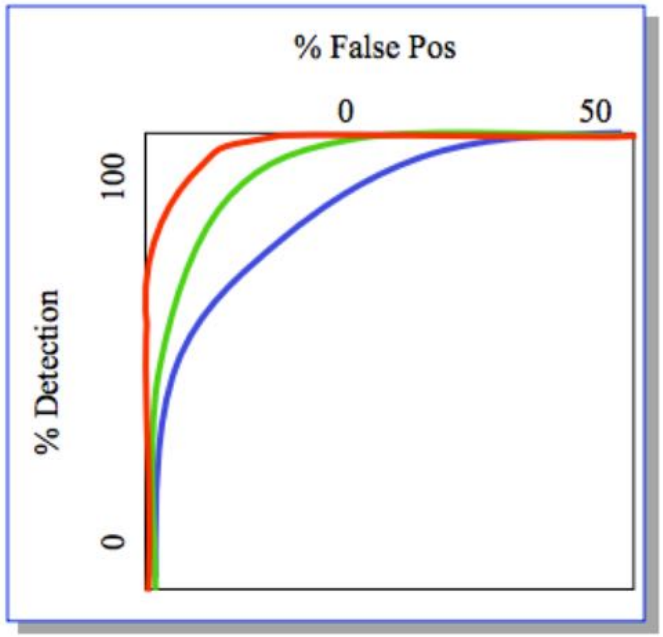


- Negativos são eliminados imediatamente!

Cascade of Boosted

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

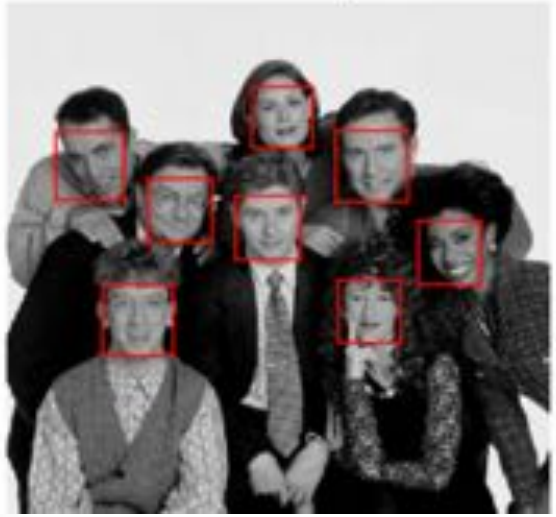


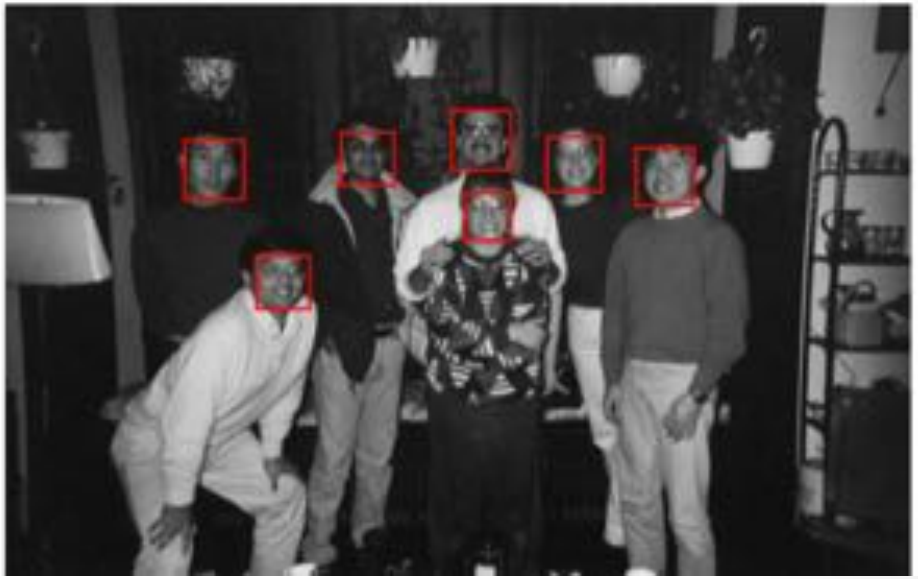


O processo de treinamento Cascading

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



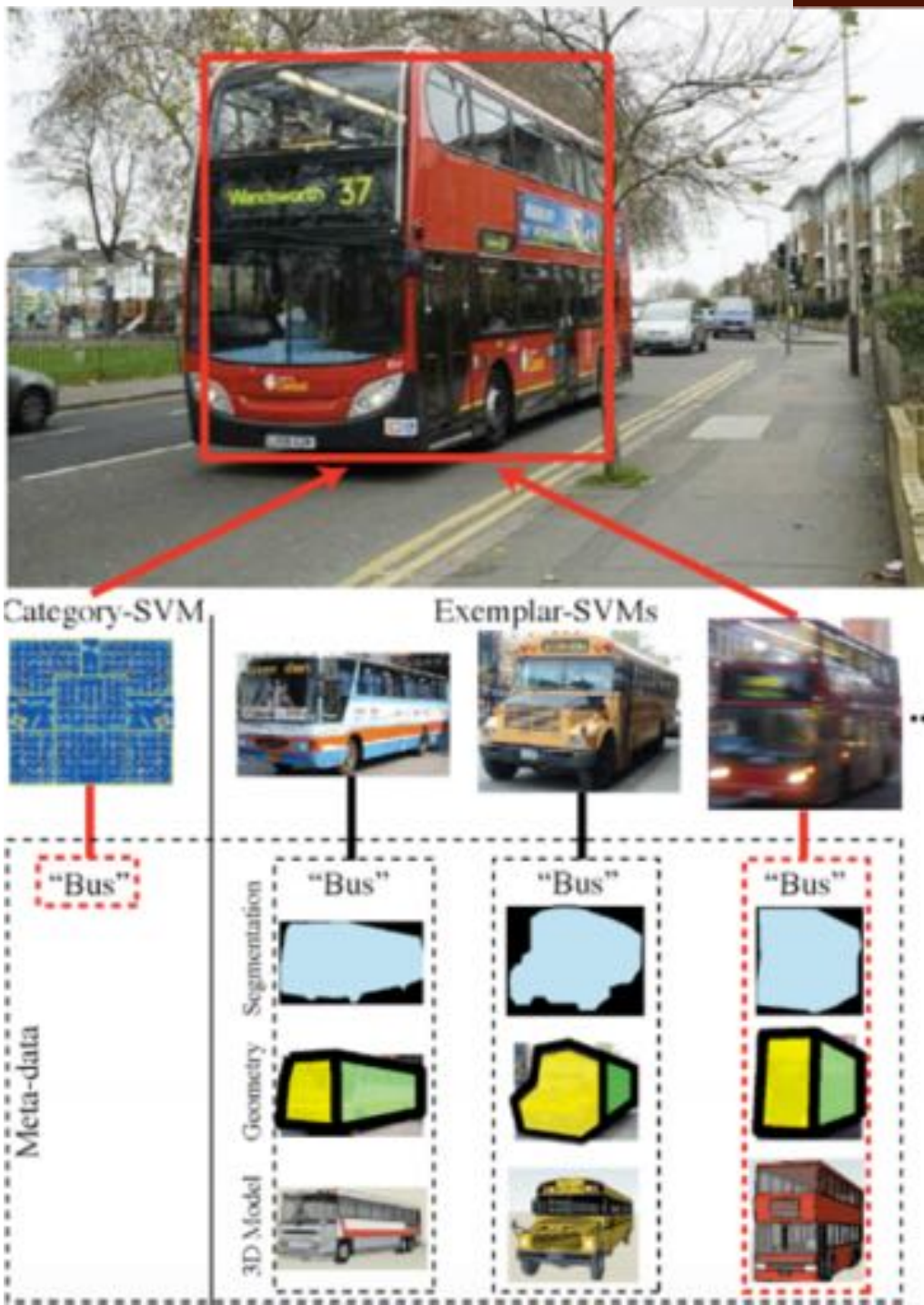


Outras abordagens?

Exemplar SVMs

Ensemble of Exemplar SVMs for Object Detection and Beyond

Tomasz Malisiewicz, Abhinav Gupta, Alexei A. Efros



A ideia

- Identifica um objeto, ao invés da classe
- Um classificador para cada exemplar
  - Sim, isso mesmo!

- Realiza a classificação eliminando o que não é o objeto em relação ao que é

- Baseado em HOG

7x4 HOG

4x8 HOG

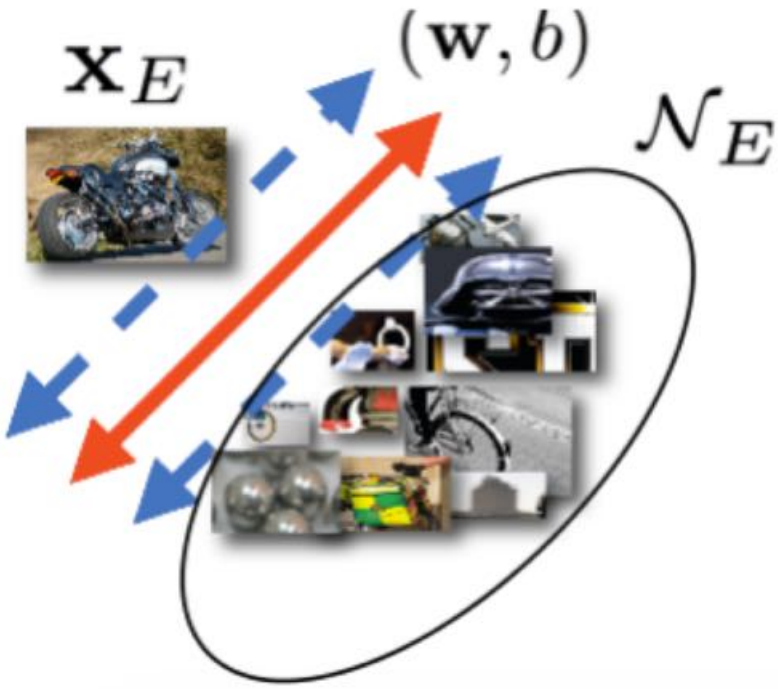
Como funciona o treinamento

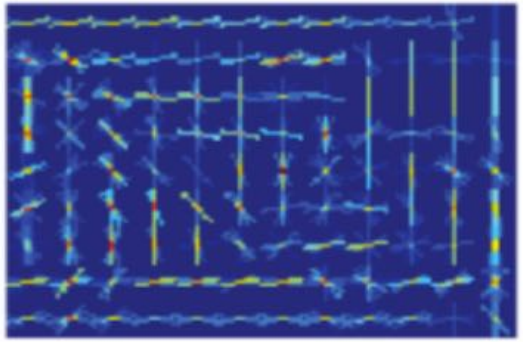
- Tire o HOG de um exemplar E (x_e)
  - Cada exemplar possui definição de dimensão própria para manter a razão de aspecto
  - normalmente 8x8 células, com aproximadamente 100 pixels
- Colete vários objetos de classe negativas, sem categoria pré-determinada (N_e)

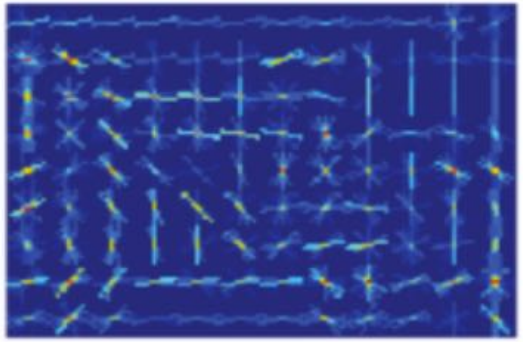
- **Gere um classificador que maximize a separação de X_e em relação a todos os N_e**

SVM lineares!

- Treina um SVM linear para cada X_e.
- O vetor de aprendizado W_e deve maximizar a margem







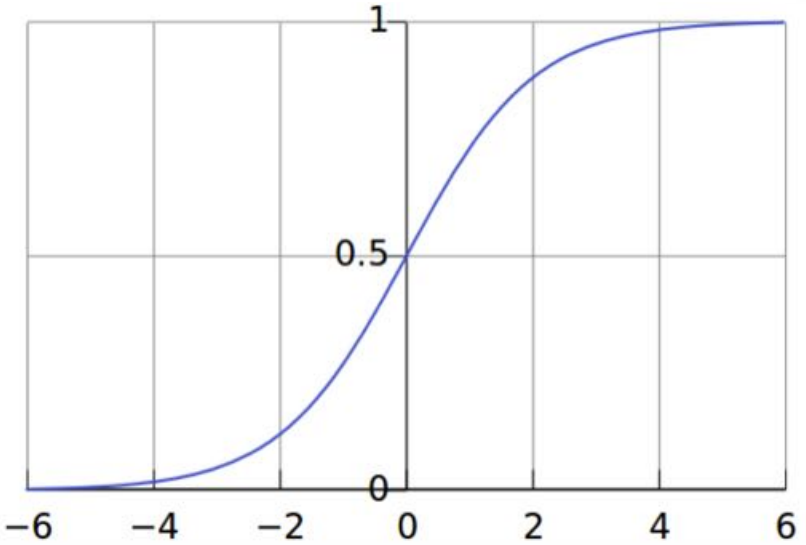
Melhores lineares

- Interativamente deixe na base de negativos apenas os melhores negativos (**hard ones**)

1. Treine
2. Encontre objetos onde estão tendo muito problemas
   - **próximo a margem**
   - **sem classificacão**
3. Deixe na base somente esses elementos
4. Treine novamente, e itere

Calibre

- Existem portanto múltiplos SVMs, um para cada exemplar
- Necessário ajustar o comite (ensemble) para dar uma resposta única
- **Baseado no fitting sigmoidal com uma base de validação positiva e negativa!**



Calibre

- Una os exemplares (SVMs deles) que posuem similaridade (como se unisse una os W_e)

>0.5 ➔ positivo

<0.2 ➔ negativo

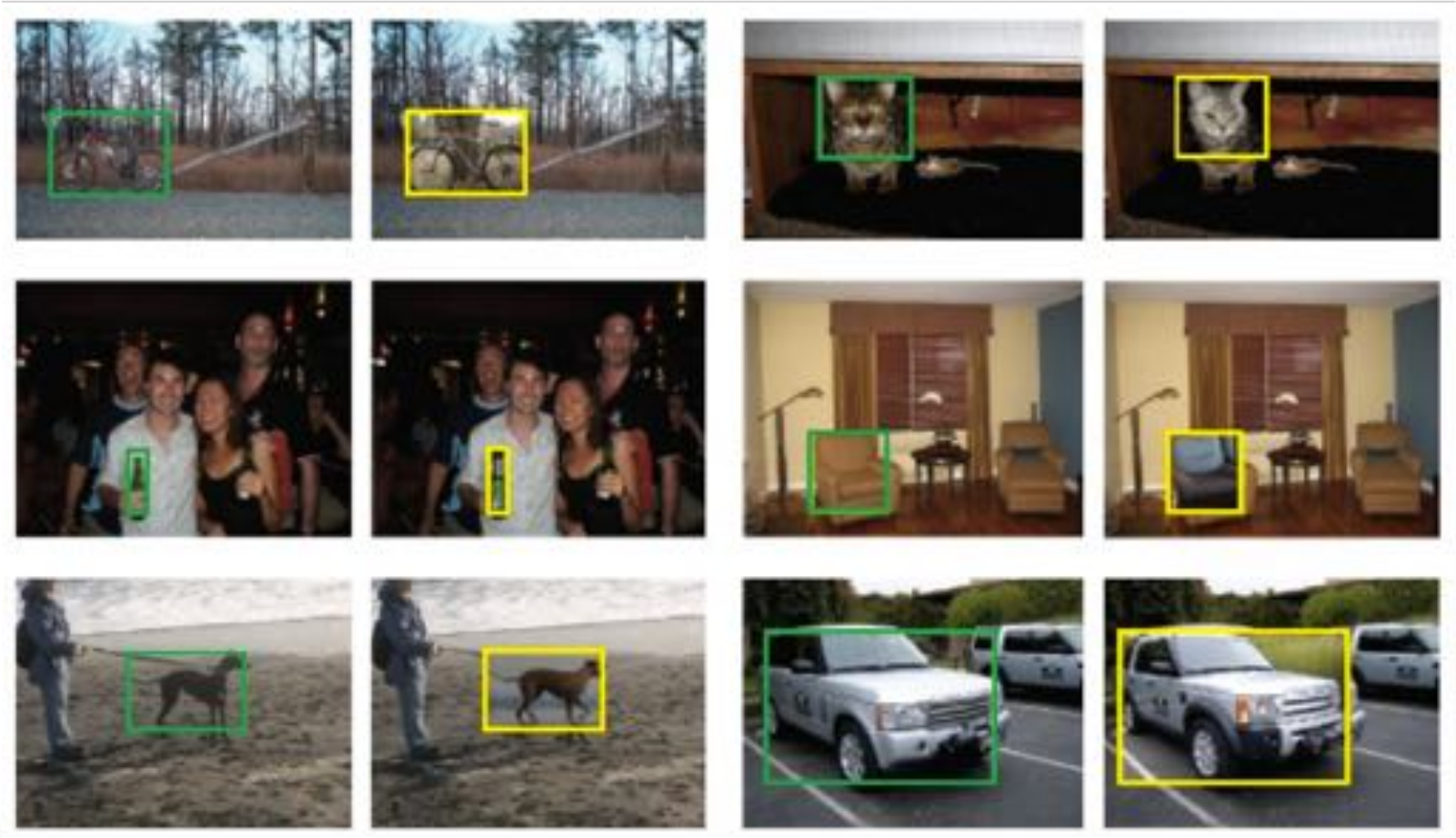
Agora fazendo a deteção

- Janela deslizante sobre a imagem (múltiplas resoluções)
- O score do svm sobre cada janela dirá o quão parecido a região é do exemplar

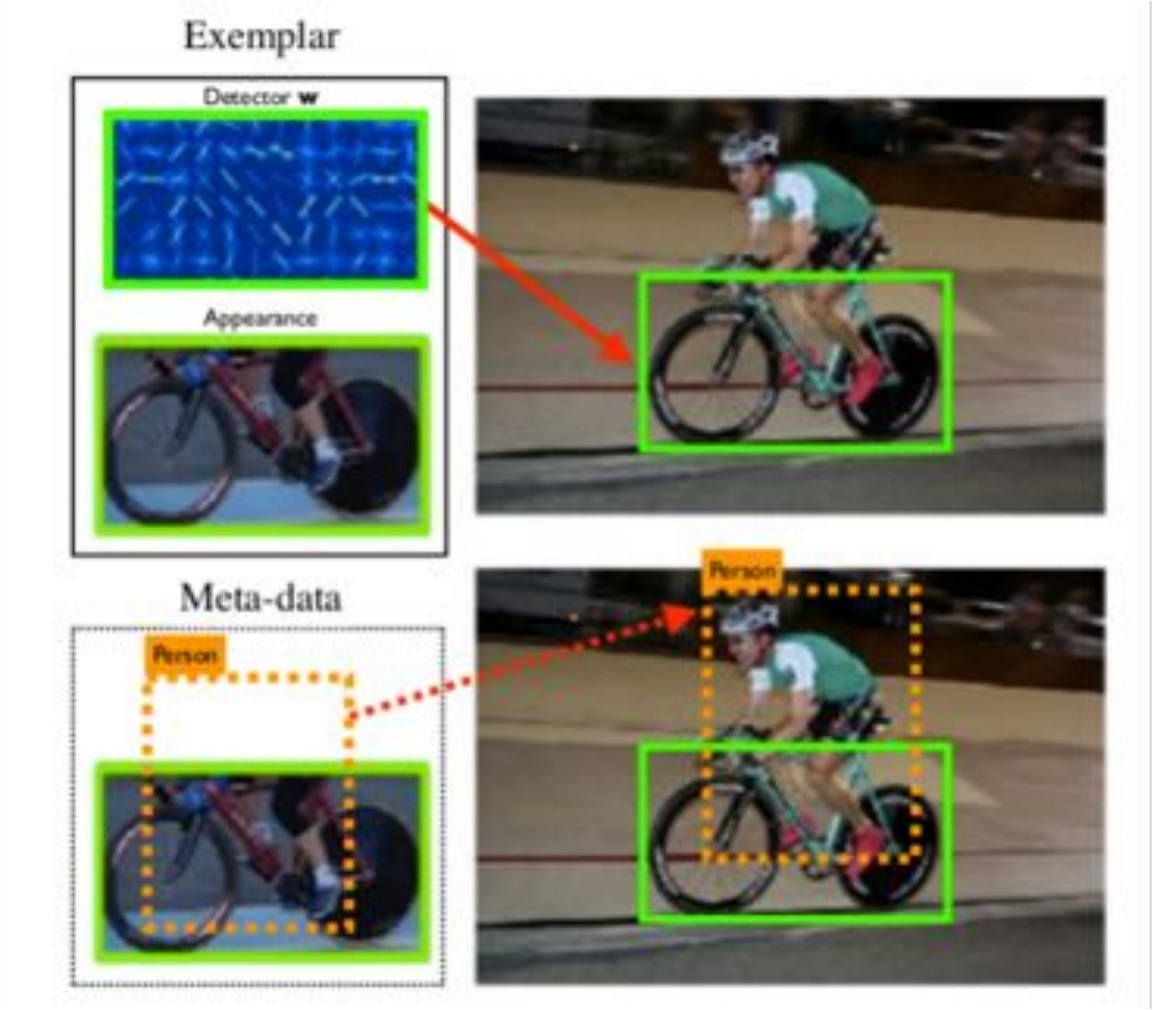
- No fim, cada janela possuirá uma nota em relação aos exemplares

Alguns resultados

Mais alguns resultados



Alguns resultados (estimação de pose)



Finalmente

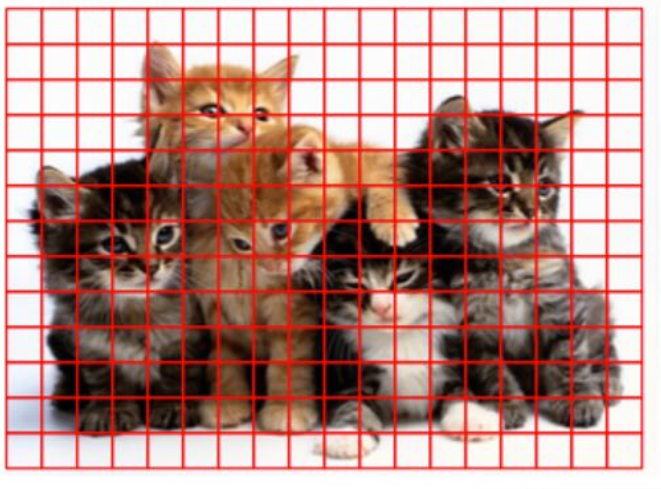
- Bom resultado
- Facilmente paralelizável
- Não necessita de aprendizado online

- Mas, muito caro computacionalmente!

Alguma abordagem com o custo menor?

- Problema das abordagens anteriores está no tempo de treinamento
  - Alto

- Como fazer com que o tempo seja menor?

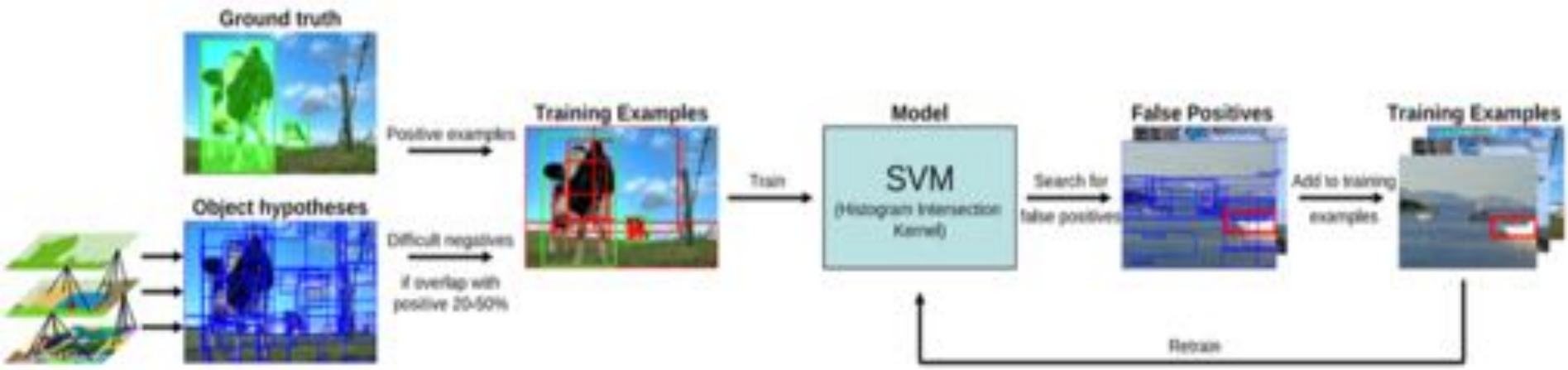


Ideia

- Se segmentarmos corretamente a imagem e depois rodar o reconhecedor do objeto
- Custo menor!

- O problema é como segmentar mantendo todas as propriedades

E o reconhecimento ficaria



Selective Search for Object Recognition

J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders

O reconhecimento

Usando uma base de treinamento e validação

1. Obtenha as propostas
2. Treine um SVM com as melhores
3. Encontre falso positivos na resposta
4. Insira falso positivos no treinamento
5. Repita

E para detectar com maior precisão:

- **Use Selective Search**
  - Codifica as regiões da imagens em sucessivas regiões melhoradas
  - Como uma hierarquia (abordagem bottom-up)

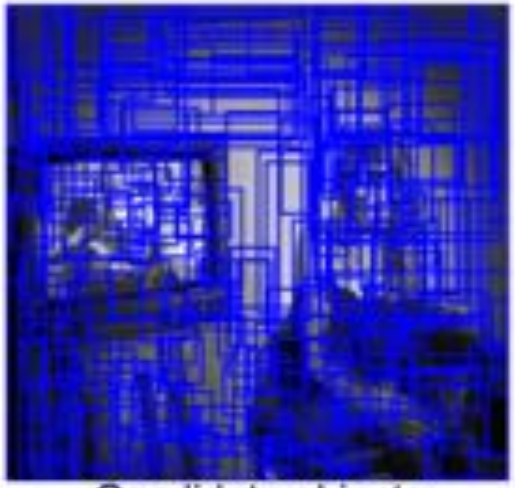
- Objetivos
  - velocidade
  - independencia de escala
  - várias informações de agrupamento

Selective Search

1. Gere os candidatos iniciais
   - Use o algoritmo de P. F. Felzenszwalb and D. P. Huttenlocher. “Efficient Graph-Based Image Segmentation.” IJCV, 59:167–181, 2004.]

Input Image

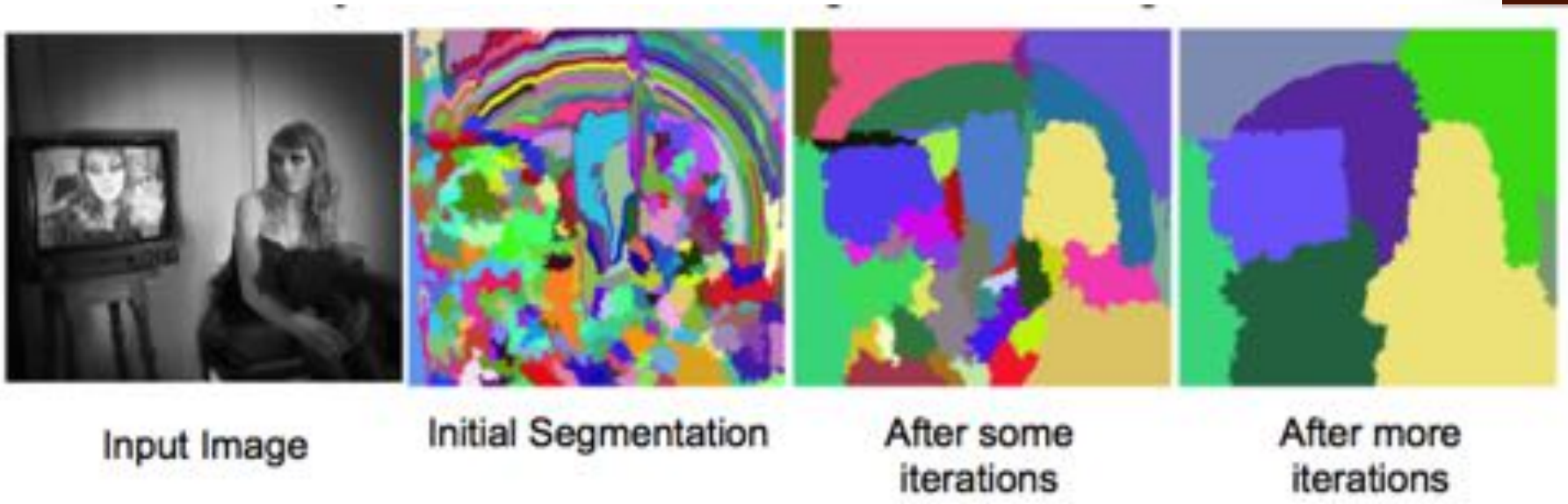
Segmentation



Candidate objects

Selective Seach

2. Combine partes similares recursivamente
   - Selecione as duas regiões mais similares
   - combine-as em uma
   - repita até que exista apenas uma região



Selective Search

3. Gere os candidatos!

E como se considera duas regiões parecidas?

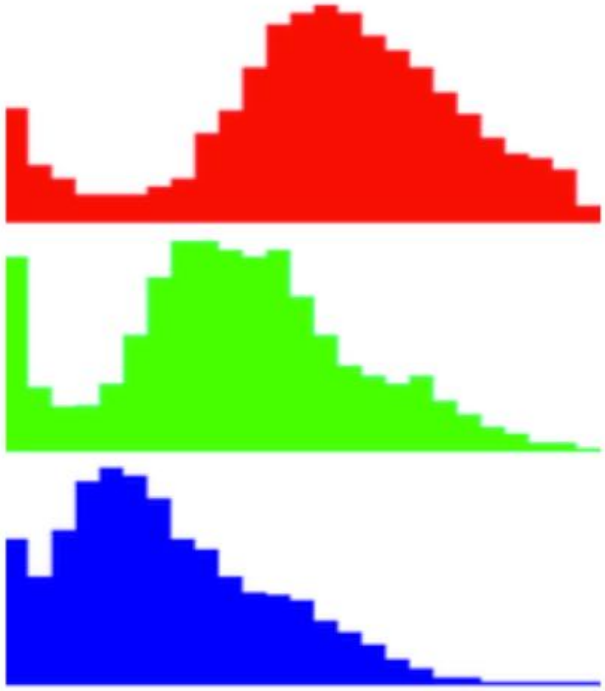
- No artigo são definidas algumas méticas
  - Mas pode-se customizar de acordo com a necessidade

- Similaridade
  - Cor
  - Textura
  - Tamanho
  - Forma

Similaridade de Cor

- Medida de similaridade de cor calculada pela através da interseção dos histogramas de cada canal

$$s_{colour}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k)$$



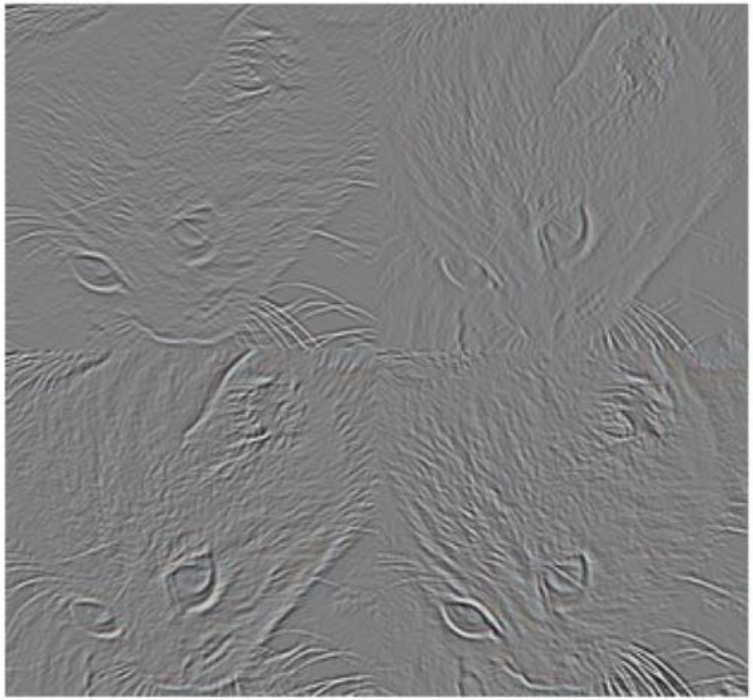
- No artigo foram usados

25 bins, portanto 75 features

- Escolha o melhor esquema de cor

Similaridade de Textura

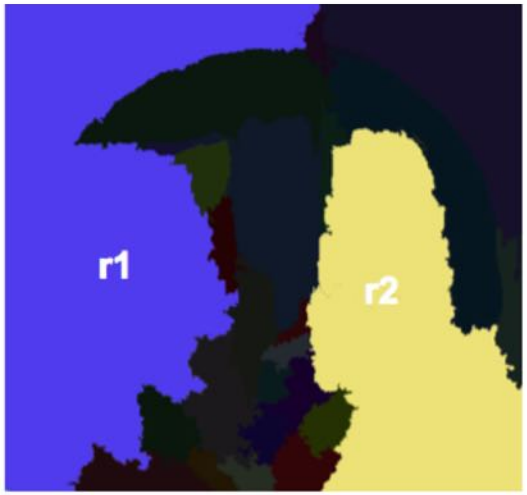
- Pode ser calculada usando HOG (original do artigo)
  - HOG de oito direções para cada canal (24 HOG)
  - Cada HOG com 10 bins (240 features)

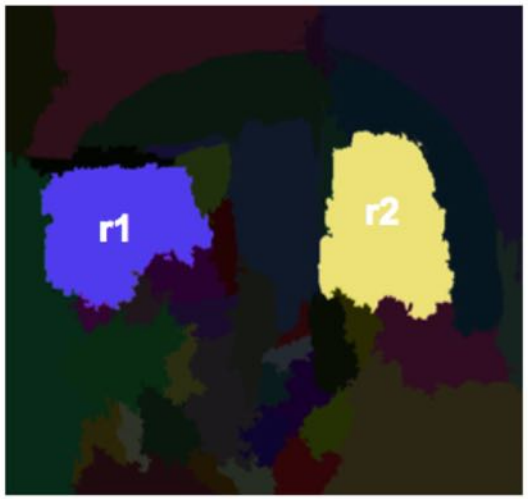


Similaridade de Tamanho

- Consiste em agrupar regiões menores com regiões maiores
- Assim, as regiões vão crescendo e se tornando balanceadas

$$s_{size}(r_i, r_j) = 1 - \frac{\text{size}(r_i) + \text{size}(r_j)}{\text{size}(im)}$$

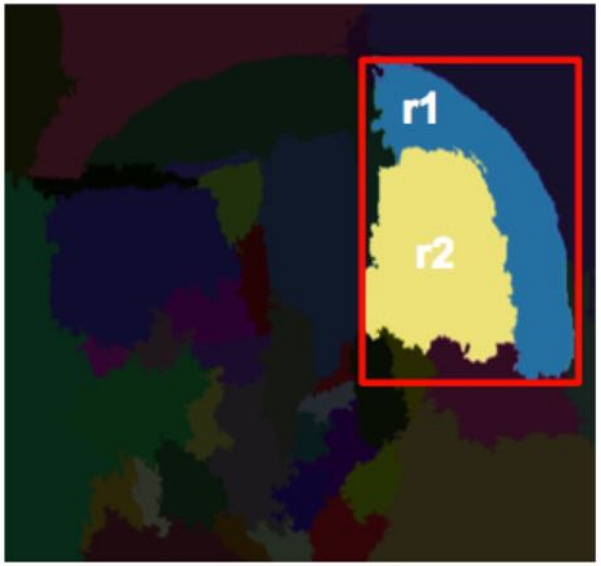




Similaridade de Forma

- Medida que avaliar o quão bom duas regiões estariam se estivessem juntas
  - Basicamente mede se existem mudanças drásticas de centro de massa

$$fill(r_i, r_j) = 1 - \frac{\text{size}(BB_{ij}) - \text{size}(r_i) - \text{size}(r_i)}{\text{size}(im)}$$



Similaridade Final

- A contribuição de cada similaridade é adicionada numa equação linear

$$s(r_i, r_j) = a_1 s_{colour}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j),$$

- Comum:
  - Usar esquemas de ponderação
  - Usar LDA para ajustar os pesos automaticamente

Alguns Resultados

(a) Bike: 0.863

(b) Cow: 0.874

(c) Chair: 0.884

(d) Person: 0.882

Conclusões

- Selective Search apresenta um modelo rápido para detecção de objetos
  - Também é eficiente
  - e customizável de acordo com o problema

- O janelamento ainda é adequado quando não se consegue uma segmentação minimamente adequada!