Face Recognition Using Boosted Local Features – Viola y Jones

Resumen
En este articulo, presentamos un nuevo metodo para el reconocimiento de rostros. Se describe un conjunto de caracteristicas, rectangulos computacionalmente eficientes, los cuales actuan en pares de imagenes de entrada. Las caracteristicas se comparan con regiones en la imagen de entrada a diferentes localizaciones, escalas y orientaciones. El algoritmo AdaBoost se aplica para entrenar la funcion face similarity mediante la selección de caracteristicas.

Introduccion

El metodo presentado establece la definicion de un nuevo conjunto de caracteristicas (image features) y la inclusion de un nuevo algoritmo de aprendizaje.

El enfoque general, es la construccion de una funcion denominada face similarity function el cual se evalua sobre dos imagenes con rostros (que han sido, cortados, normalizados en traslacion, escala y rotacion). Esta funcion se aprende a partir de una base de datos. Esta funcion puede ser umbralizado para permitir una decision binaria. La funcion face similarity consiste de una combinacion lineal de features rectangles. Estas caracteristicas han sido modificadas para aplicarse a un par de imagenes de entrada o como tambien a una imagen.

Filtros, Caracteristicas y Clasificadores

El problema de reconocimiento de rostros esta relacionado a los problemas de reconocimiento y verificacion. Ambos problemas asumen una galeria de rostros con identidades conocidas. Ambos problemas pueden ser resueltos mediante una funcion denominada face similarity function. Esta funcion toma dos imagenes como entrada y la salida es una medida de su similaridad.

$$F(I_1, I_2) \in \mathcal{R}$$

Donde i1 e I2 son imagenes de entrada que han sido cortados y normalizados.

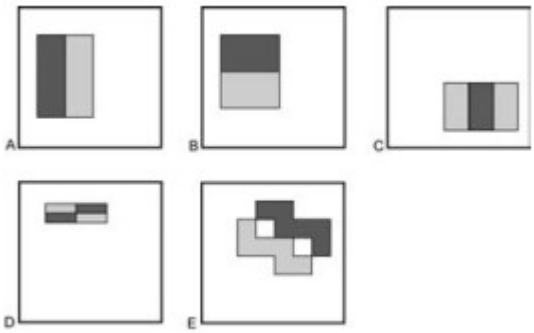
Viola y Jones presentan a la funcion de face similarity como una suma de caracteristicas

$$F(I_1, I_2) = \sum_{i=1}^{N} f_i(I_1, I_2).$$

Una caracteristica consiste de un filtro que actua sobre ambas imagenes de entrada.

$$f_i(I_1, I_2) = \begin{cases} \alpha & \text{if } |\phi_i(I_1) - \phi_i(I_2)| > t_i \\ \beta & \text{otherwise} \end{cases}$$

Donde ti es real, una caracteristica de umbral y phi(i) es el filtro, como es una funcion escalar adoptamos filtos lineales. Los filtros usados se muestran en la siguiente figura



El filtro se calcula sumando la intensidad de todos los pixeles en la region oscura y quitandola de la suma de los pixeles en la region clara.

El calculo de los filtros se acelera grandemente usando el concepto de imagen integral, aplicado a las imagenes de entrada.

Algoritmos de Aprendizaje

Nosotros usamos una version mejorada de AdaBoost, derivado del trabajo de Schapire y Singer, que usa confidence-rated-predictions. Esta version de AdaBoost simplifica la notacion y permite un analisis simplificado, tambien, para el manejo de grandes bases de datos, nosotros modificamos el algoritmo usando el concepto de remuestreo.

Ada Boost
Acorde al trabajo de Schapire ans Singer, el algoritmo AdaBoost asigna a cada ejemplo xi un peso Di, donde Di se inicia en 1/N, siendo N el total de muestras. La correcta etiqueta para cada muestra en yi es +1 o -1

La hipotesis es:

$$h_j(I_1, I_2) = |\phi_j(I_1^i) - \phi_j(I_2^i)|$$

entonces

$$f_j(x_i) = f_j(I_1^i, I_2^i) = \begin{cases} \alpha & \text{if } h_j(I_1^i, I_2^i) > t_j \\ \beta & \text{otherwise} \end{cases}$$

En cada ronda, escogemos un clasificador el cual minimiza el error:

$$\epsilon_j = \sum_{\substack{i\,:\,y_i = +1 \\ \wedge\, h(x_i) \leq t}} D_i \quad + \quad \sum_{\substack{i\,:\,y_i = -1 \\ \wedge\, h(x_i) > t}} D_i$$

Los primeros terminos son la suma de los pesos de las muestras que generan falsos negativos de fj y el segundo termino es la suma de los pesos de las muestras que son falsos positivos. Minimizando la suma de estos terminos minimizamos el error weighted

Los valores de alfa y beta se calculan según:

$$\Rightarrow \alpha = \frac{1}{2} log\,(\frac{W_+^+}{W_-^+})$$

$$\beta = \frac{1}{2} log\,(\frac{W_+^-}{W_-^-}).$$

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where the labels $y_i \in \{-1, +1\}$ for negative and positive examples respectively. For face recognition, $x_i = (I_1^i, I_2^i)$.
- Initialize weights $D_{1,i} = \frac{1}{n}$ where n is the total number of negative and positive examples.
- Let R be the number of rounds to boost before resampling
- For $t = 1, \ldots, T$:

  1. Normalize the weights,

     $$w_{t,i} \leftarrow \frac{D_{t,i}}{\sum_{j=1}^{n} D_{t,j}}$$

     so that D_t is a probability distribution.
  2. For each filter, ϕ_j, compute the best weak classifier, h_j, that uses ϕ_j. This amounts to finding the optimal threshold t_j minimizing equation 6 for each possible filter. The error, ϵ_j, is defined in equation 6.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Choose α and β according to equations 9 and 10. This defines the feature, f_t given in equation 3.
  5. If t is a multiple of R then resample to generate a new training set with new weights.
     Otherwise update the weights:

     $$D_{t+1,i} = D_{t,i} e^{-f_t(x_i) y_i}$$

- The final strong classifier is:

$$F(x) = sign\left(\sum_{t=1}^{T} f_t(x)\right).$$