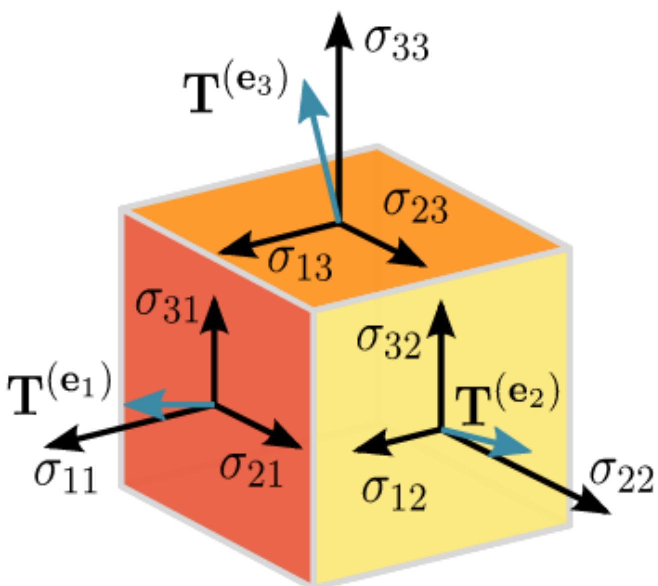
Tensor Discriminant Analysis

Image Processing - II

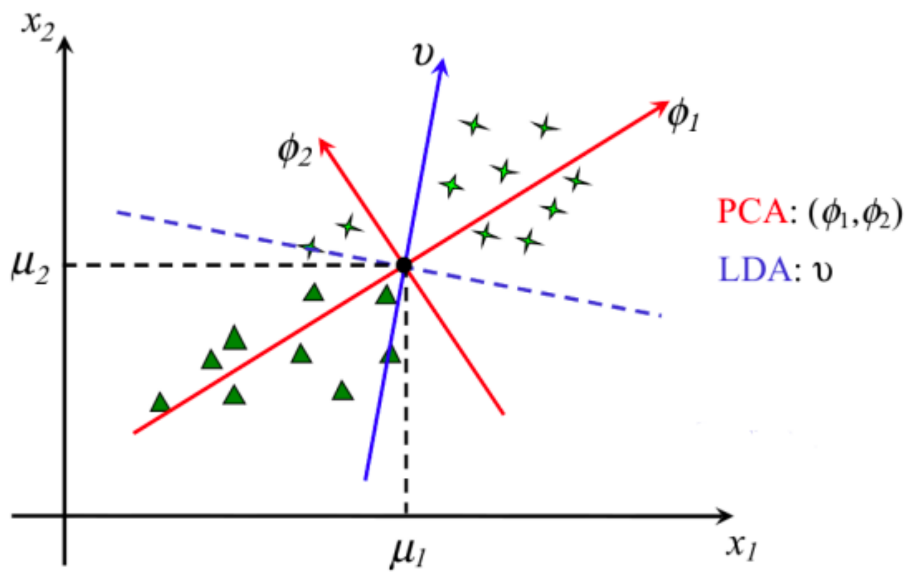
Tensor?

- Generalized linear quantity or geometrical entity that can be expressed as multi-dimensional array



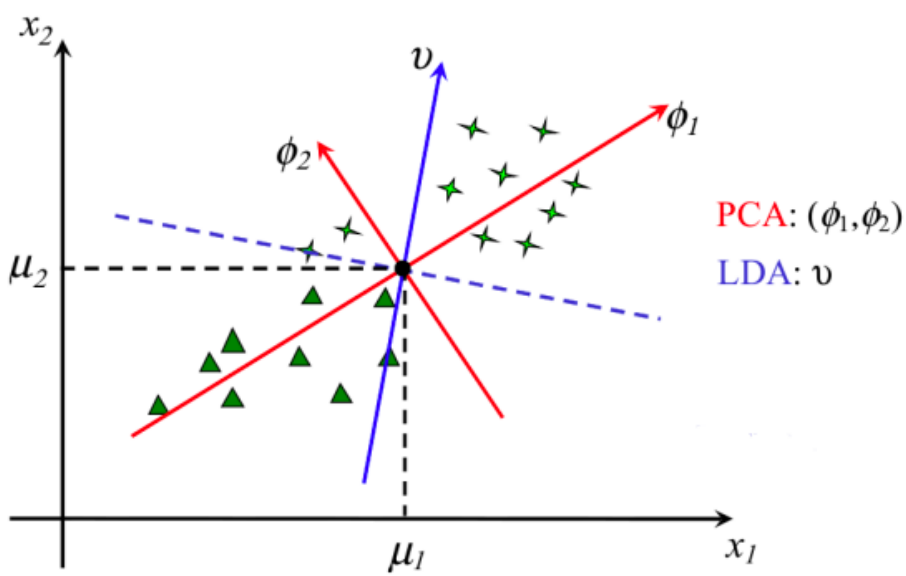
LDA?

- Finds linear combination of features which characterize two or more different objects/classes.
- Closely related to PCA

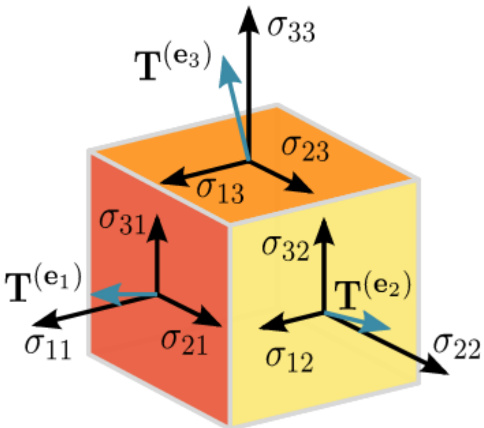


LDA?

- Following graph shows the difference between LDA and PCA



Enough about LDA!



LDA = Tensor Discriminant Analysis

Magic!!!

Separable Tensor Discriminant Analysis?

- Treat images as images not as vectors!
- Loss of features when converting to vectors
- Projection matrices are learned quickly
- Tackles small sample size problems
- Performs well in visual object detection and recognition

Algorithm:

Input: a training set $\{(\boldsymbol{\mathcal{X}}^{\alpha}, y^{\alpha})\}_{\alpha=1,\ldots,N}$ of image patches $\boldsymbol{\mathcal{X}}^{\alpha} \in \mathbb{R}^{m \times n}$ with class labels $y^{\alpha} \in \{-1, +1\}$

Output: a R-term solution of a second-order projection tensor $\boldsymbol{\mathcal{W}} = \sum_r \mathbf{u}_r \otimes \mathbf{v}_r$

---

for $r = 1, \ldots, R$

  $t = 0$

  randomly initialize $\mathbf{u}_r(t)$

  orthogonalize $\mathbf{u}_r(t)$ w.r.t. $\{\mathbf{u}_1, \ldots, \mathbf{u}_{r-1}\}$

  repeat

    $t \leftarrow t + 1$

    contract $x_k^{\alpha} = \mathcal{X}_{kl}^{\alpha} \, u_{r_k}(t)$

    compute $\mathbf{v}_r(t) = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$

    orthogonalize $\mathbf{v}_r(t)$ w.r.t. $\{\mathbf{v}_1, \ldots, \mathbf{v}_{r-1}\}$

    contract $x_l^{\alpha} = \mathcal{X}_{kl}^{\alpha} \, v_{r_l}(t)$

    compute $\mathbf{u}_r(t) = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$

    orthogonalize $\mathbf{u}_r(t)$ w.r.t. $\{\mathbf{u}_1, \ldots, \mathbf{u}_{r-1}\}$

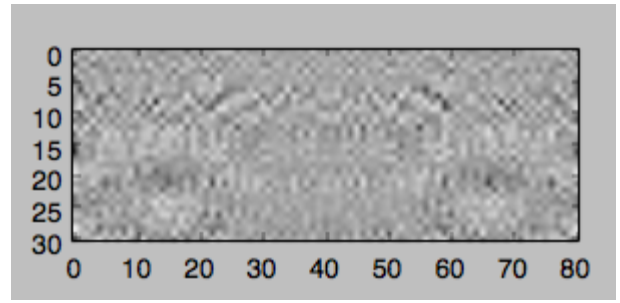
  until $\|\mathbf{u}_r(t) - \mathbf{u}_r(t-1)\| \leq \epsilon \vee t > t_{\max}$

endfor

W and Threshold

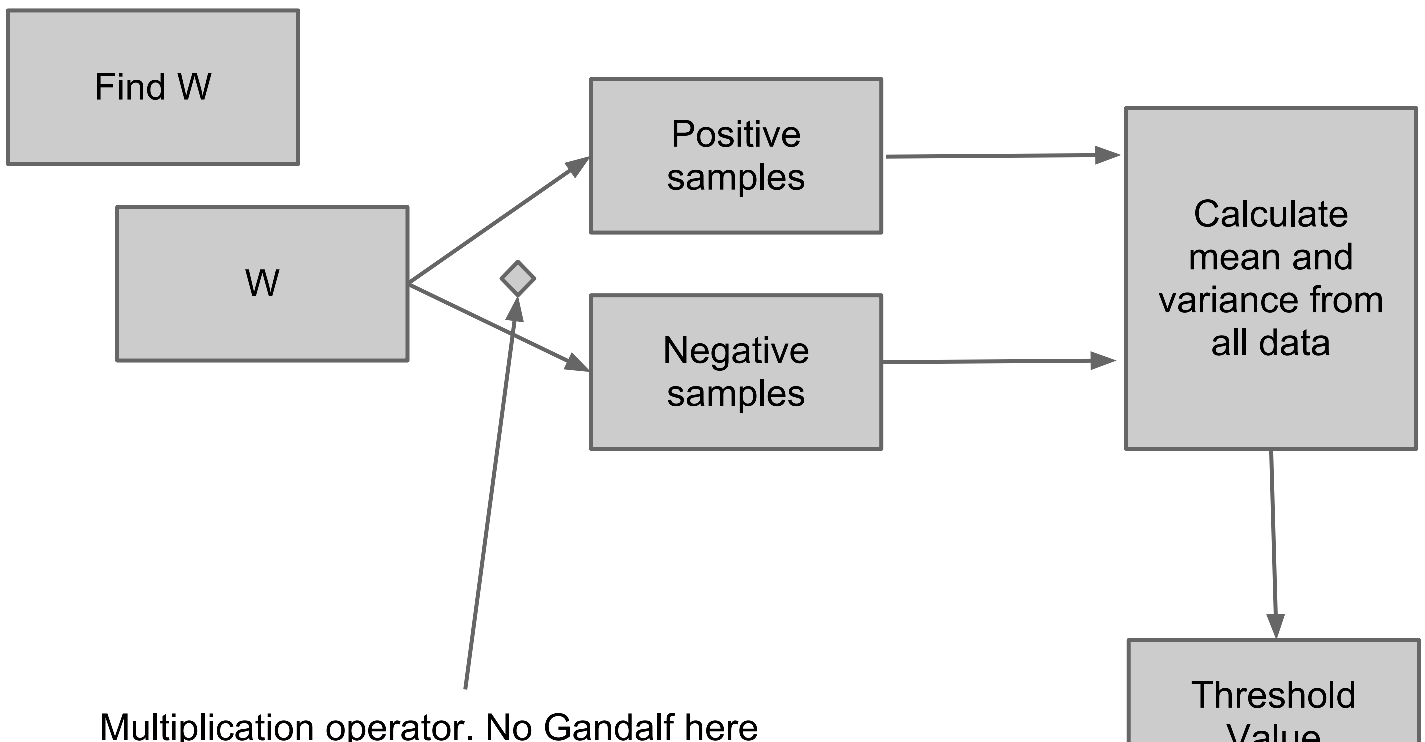
- W looks like:

(k = 9, eps = 0.0001)

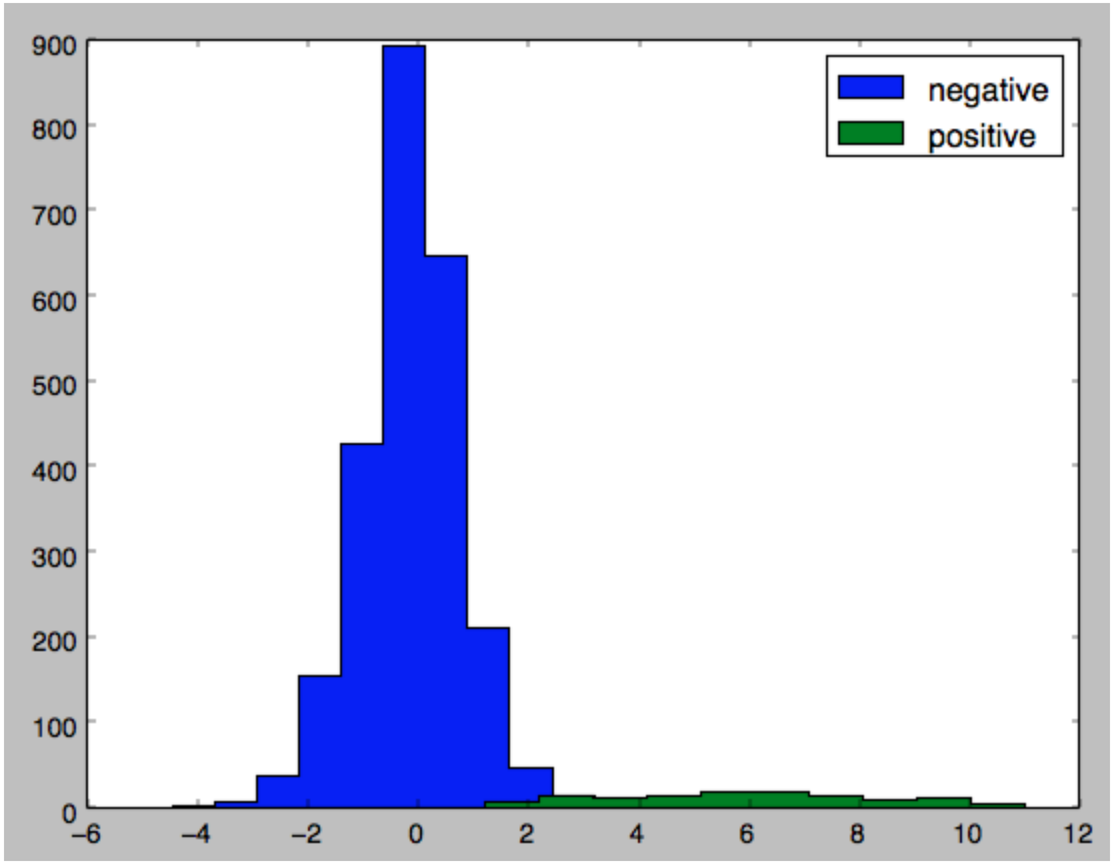


- Threshold, theta is calculated by computing the mean and variance of the projections of positive samples onto discriminant direction. (Original Paper)

What We Did?


Find W
W
Positive samples
Negative samples
Calculate mean and variance from all data
Threshold Value
Multiplication operator. No Gandalf here

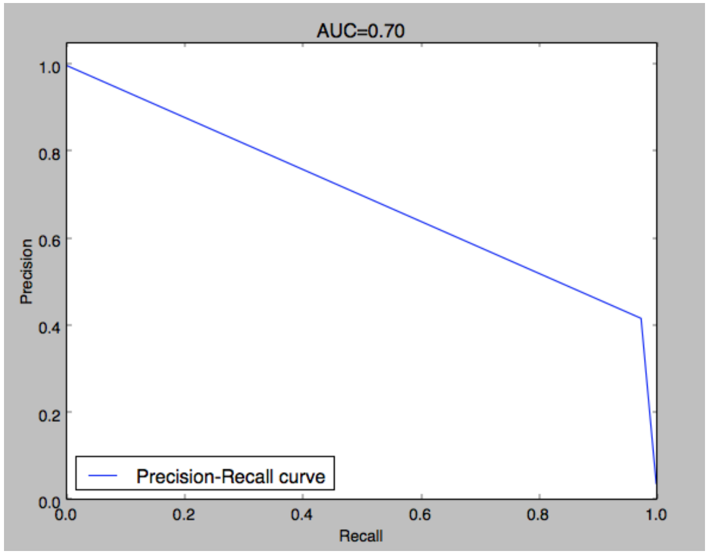
Histogram



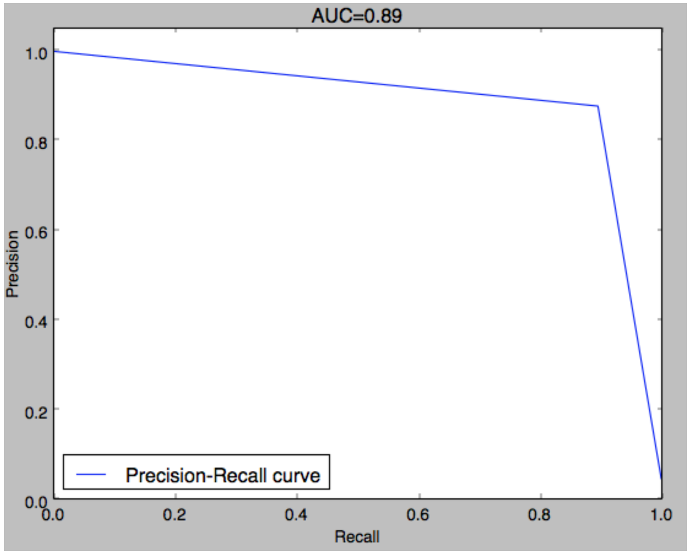
900
800
700
600
500
400
300
200
100
0
−6
−4
−2
0
2
4
6
8
10
12
negative
positive

Precision, recall and accuracy

- k = 6, theta = 1.0

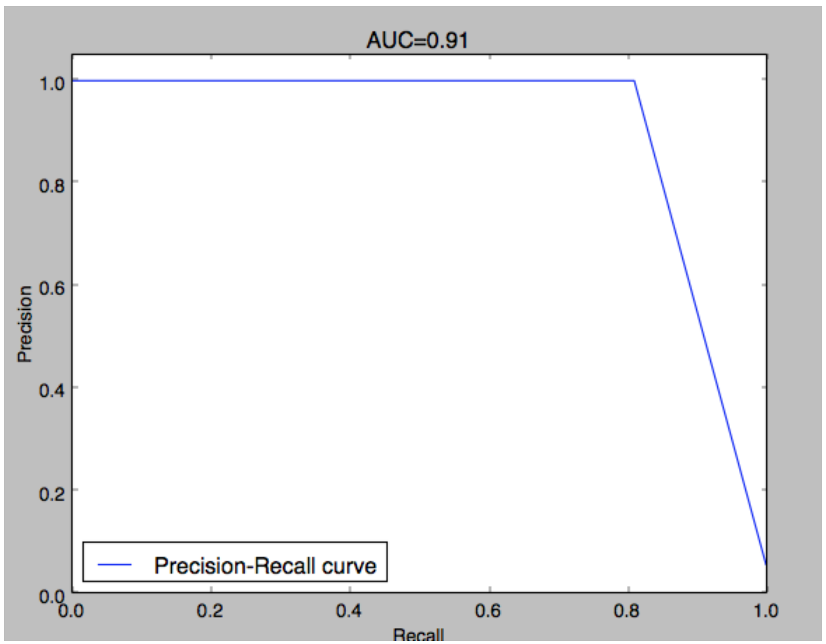


- k = 6, theta = 2.0

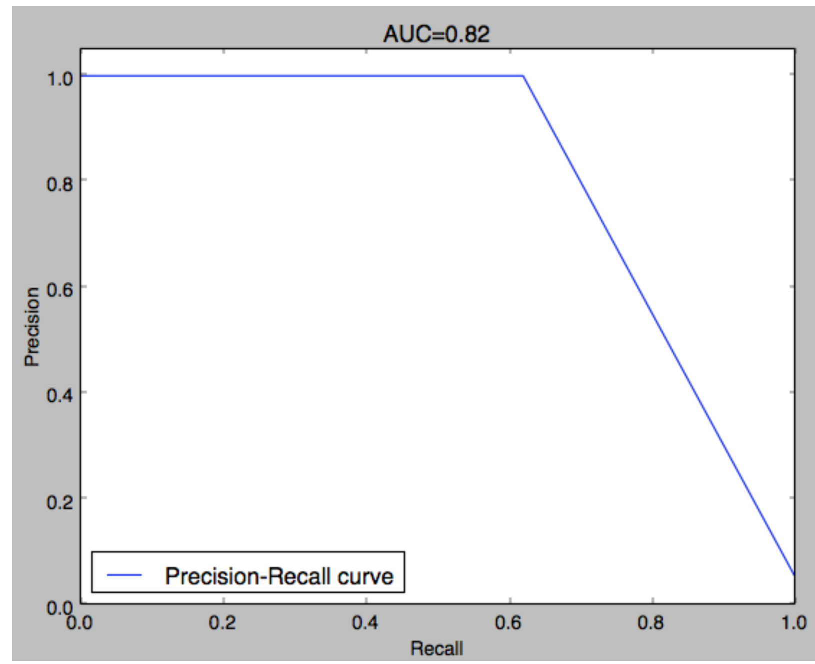


Precision, recall and accuracy

- k = 6, theta = 3.0

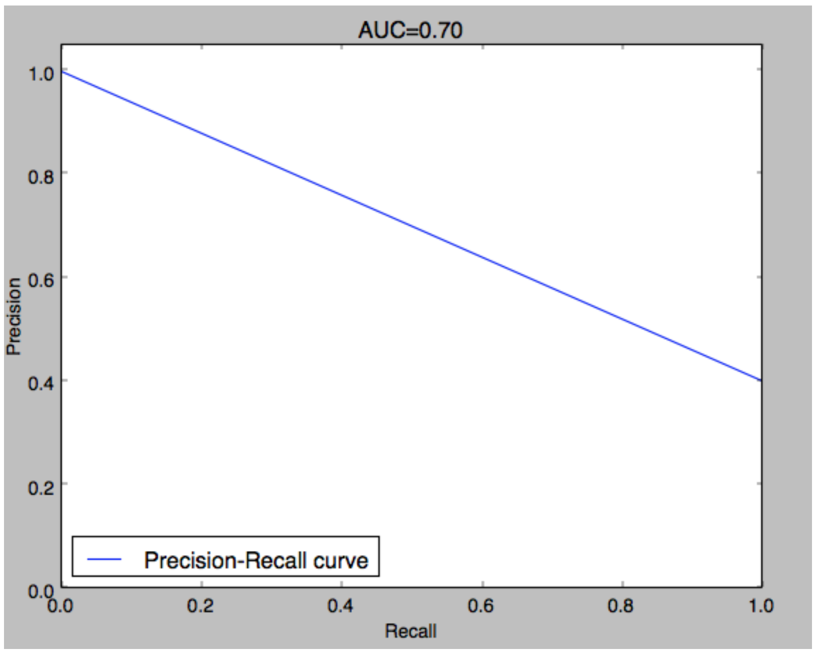


- k = 6, theta = 4.0

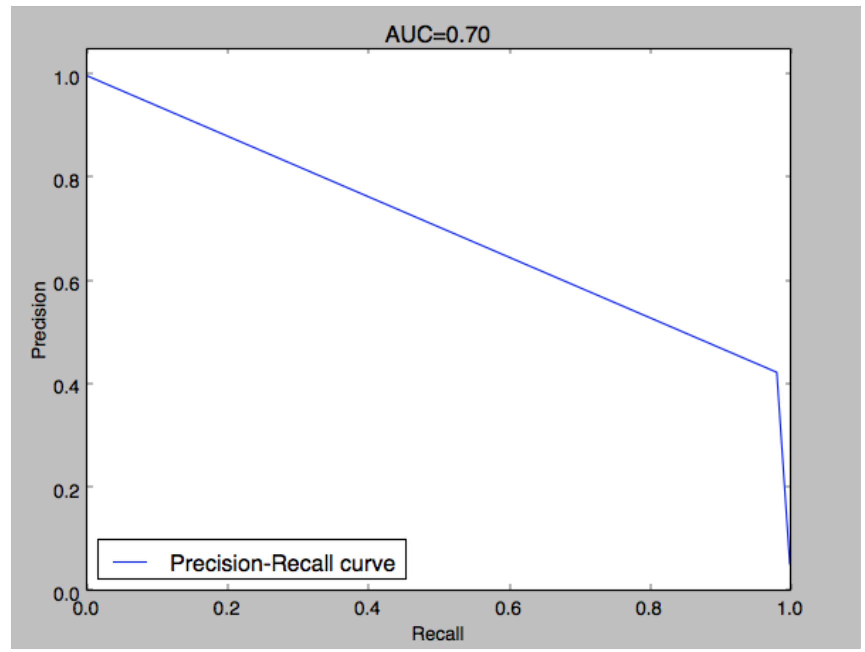


Precision, recall and accuracy

- k = 7, theta = 1.0

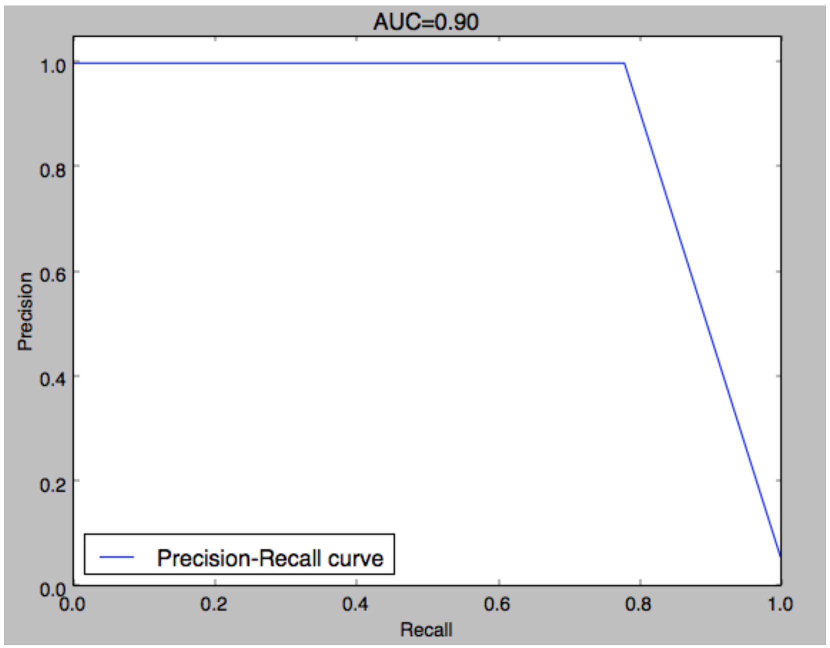


- k = 8, theta = 1.0

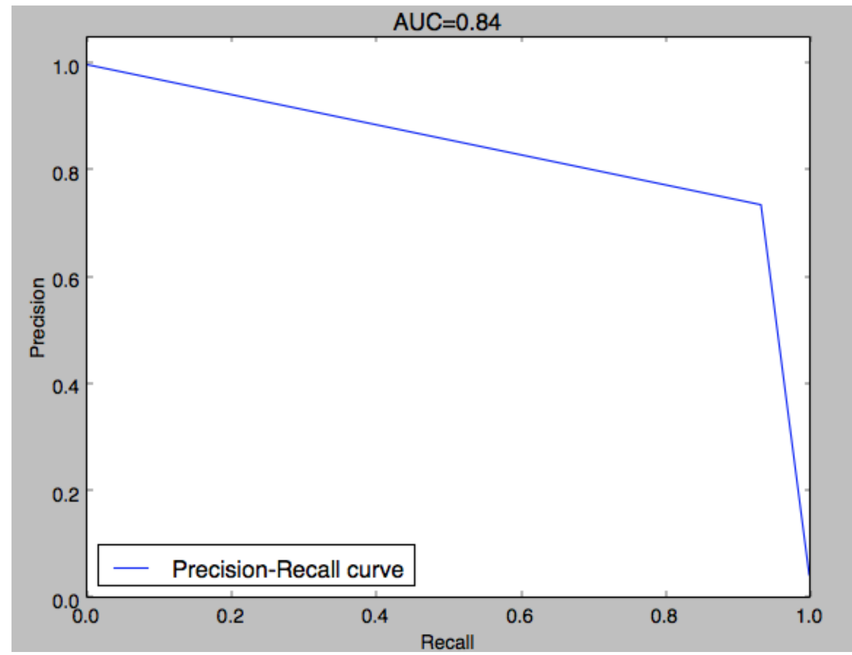


Precision, recall and accuracy

- k = 9, theta = 1.0

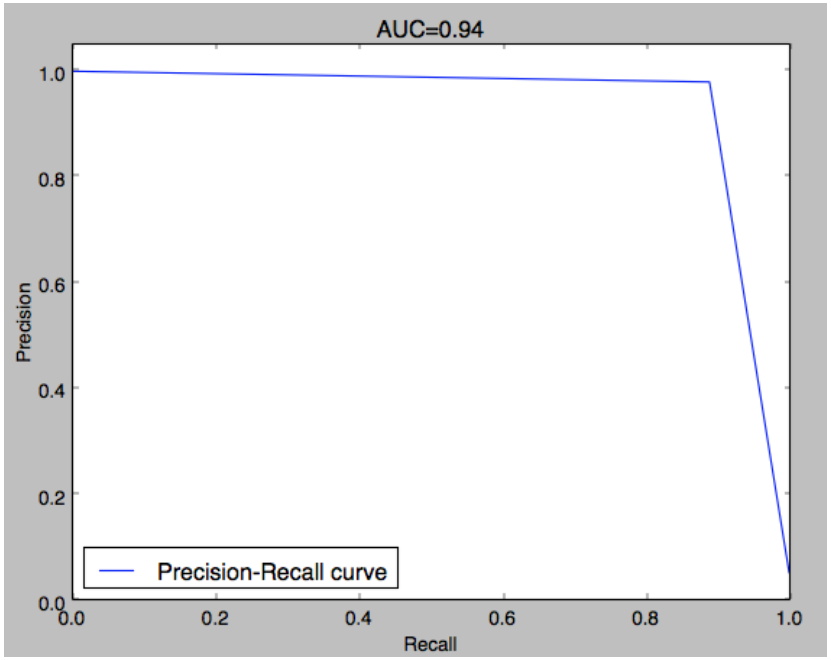


- k = 9, theta = 2.0

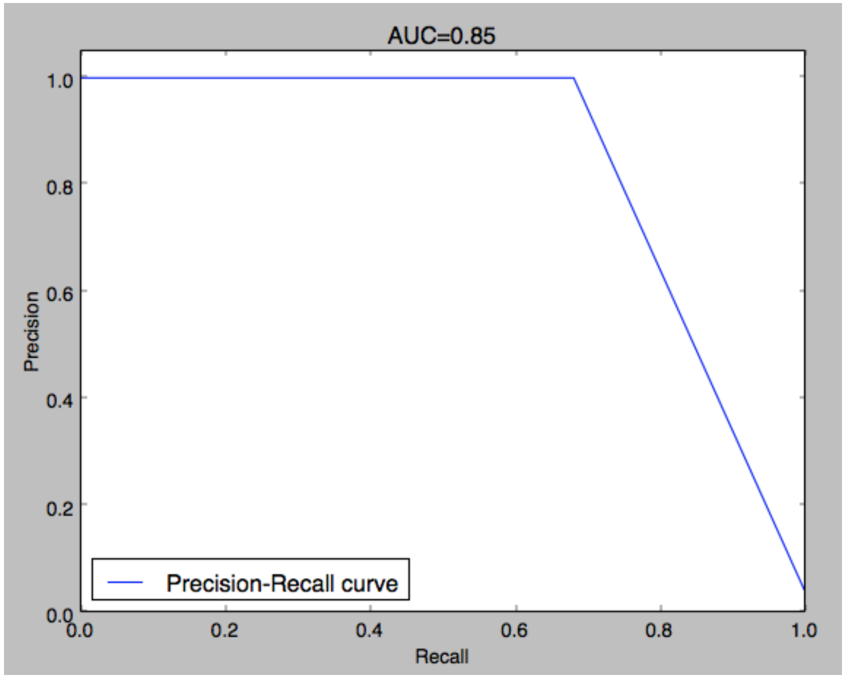


Precision, recall and accuracy

- k = 9, theta = 3.0



- k = 9, theta = 4.0



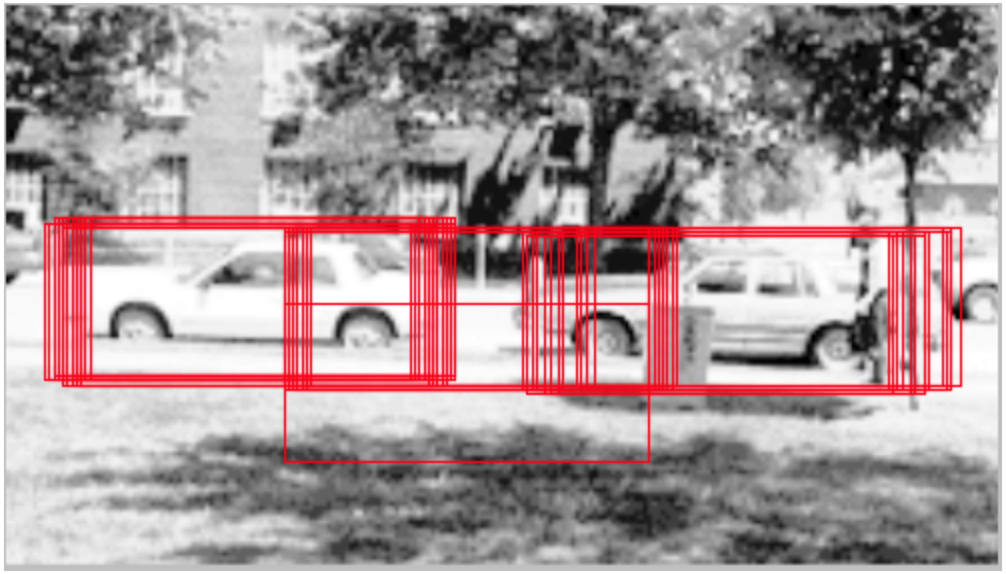
k = ?, theta = ?

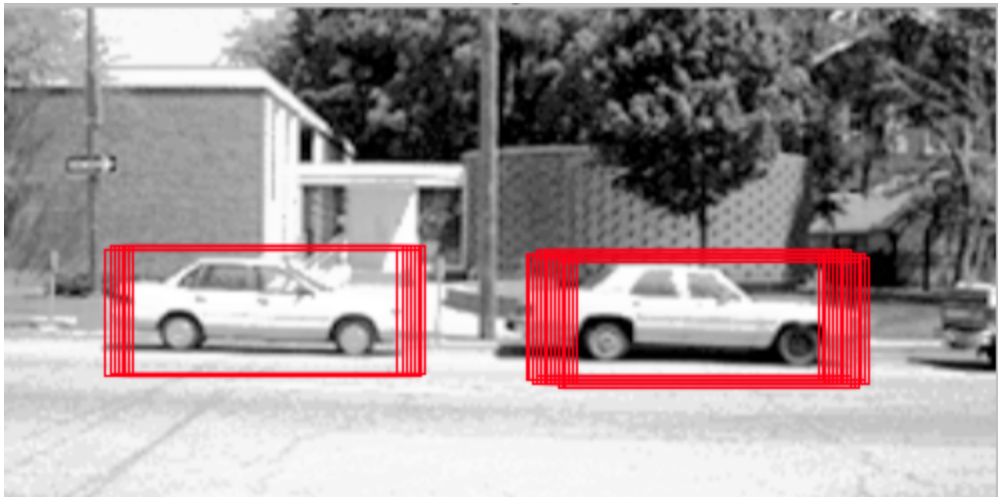
- K was chosen to be 9
- theta = 3.0
- Accuracy measure = AUC

Results on Test data

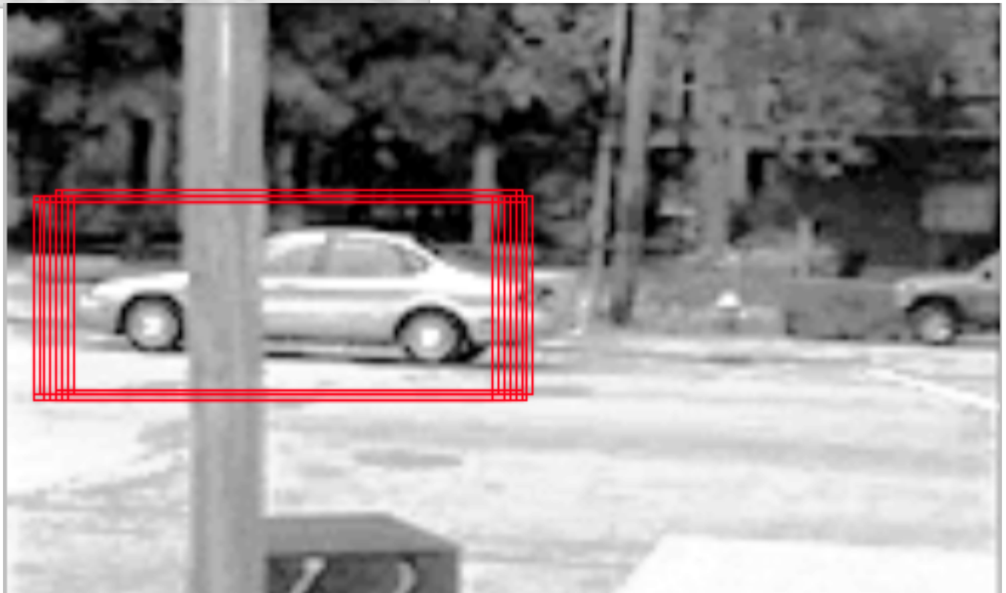
- No convolution while creating patch
- So no convolution here
- Sliding Window
- Image was multiplied by patch
- Thresholded
- Results...

Results on Test data





Results on Test data



Pretty Good huh! :D

Results on Test data