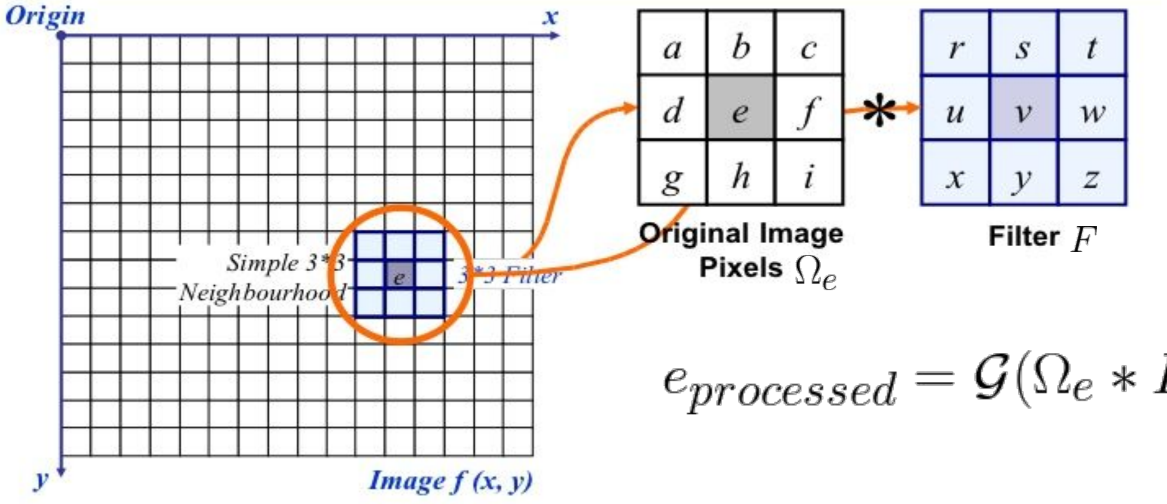
Tutorial
filtering, morphology and canny edge detection

Ziang Cheng

Outline

- Image filtering
  - Review
  - Boundary handling (padding)
  - Image morphology
- Edge detection algorithm
  - F1 evaluation metric

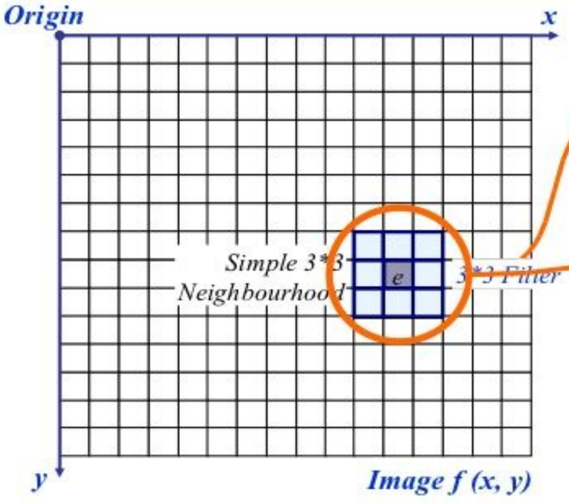
Spatial image filtering

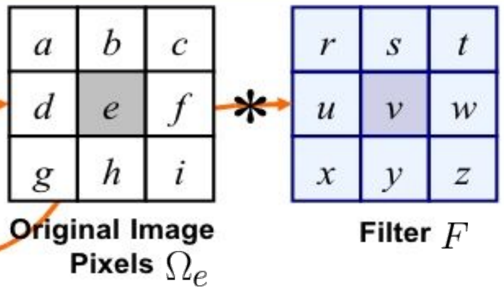


$$e_{processed} = \mathcal{G}(\Omega_e * F)$$

1. Compute the stacked response of F on Ω_e with some element-wise operator $*$
2. Gather the response to a single value through some reduction function $\mathcal{G}(\cdot)$
3. Slide the window to next pixel

Cross-correlation/convolution





$$e_{processed} = \mathcal{G}(\Omega_e * F) = \begin{aligned} &v*e + \\ &r*a + s*b + t*c + \\ &u*d + w*f + \\ &x*g + y*h + z*i \end{aligned}$$

- $*$ is element-wise multiplication operator
- $\mathcal{G}(\cdot)$ is the summation function
- $*$ is not to be confused with convolution operator *
- Processed image is linear w.r.t. input

Cross-correlation/convolution

*

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

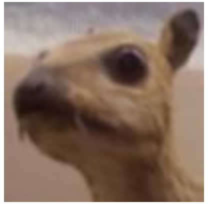
Laplacian

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpen

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Mean blur



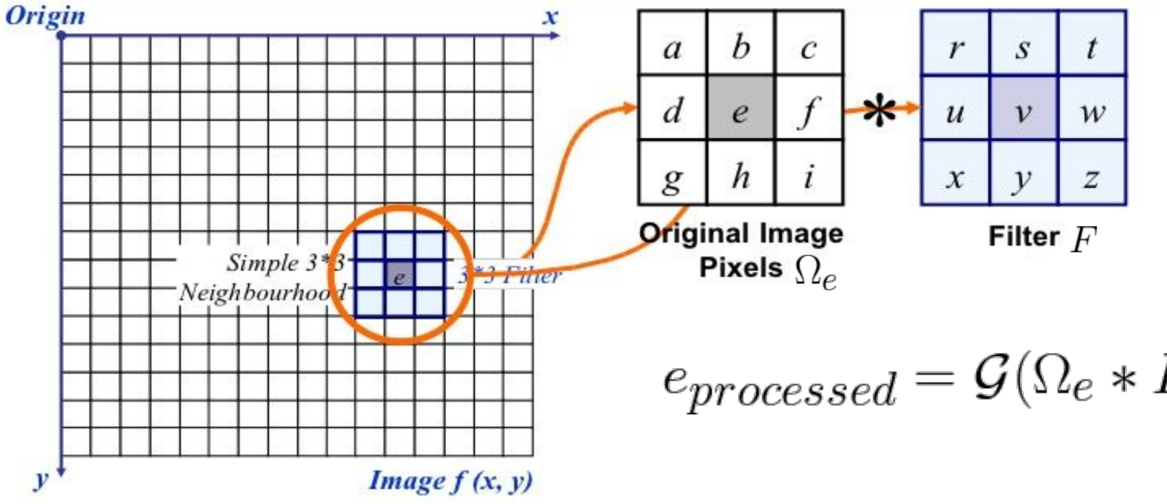
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Gaussian 3x3

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Gaussian 5x5

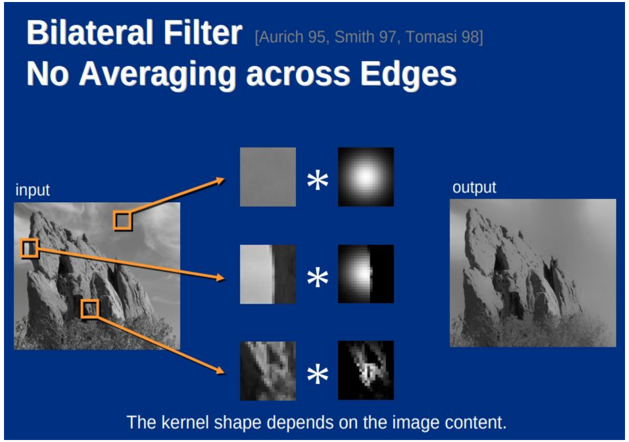
Median filter

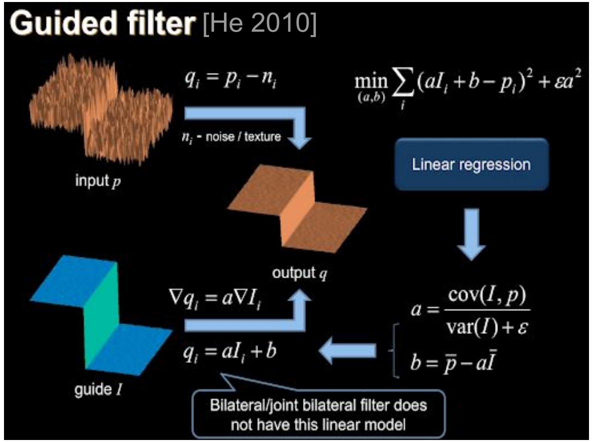


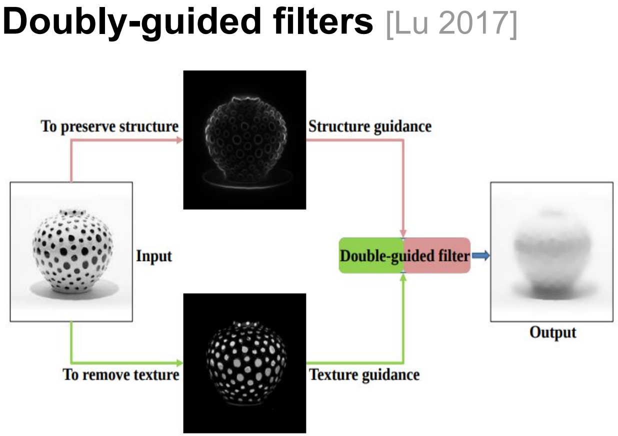
$$e_{processed} = \mathcal{G}(\Omega_e * F)$$

- $*$ is the element-wise multiplication
- $\mathcal{G}(\cdot)$ is the median function
- F is $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Filters with content-aware kernels





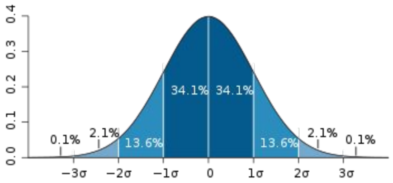


Gaussian filter (implementation)

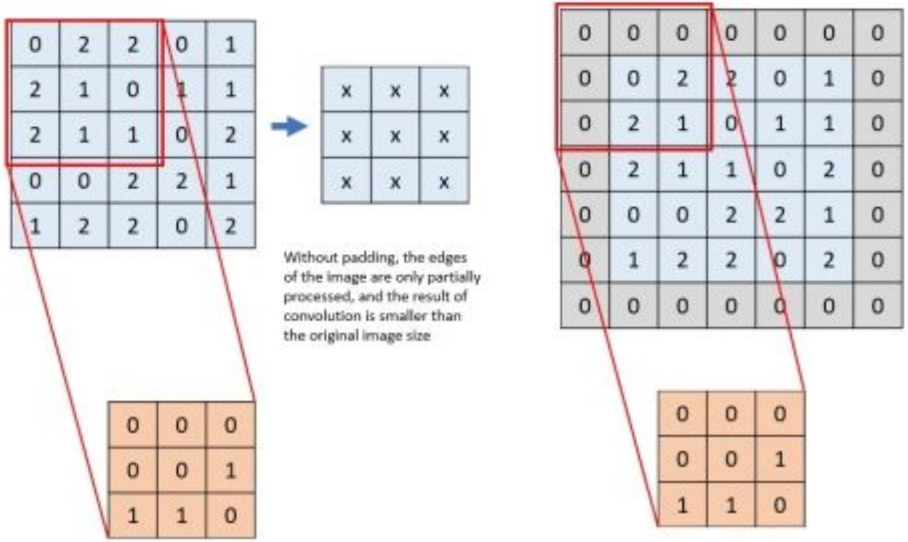
- Separability: Gauss kernels can be factorized $\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}=\begin{bmatrix}1\\2\\1\end{bmatrix}\begin{bmatrix}1&2&1\end{bmatrix}$
    - Since convolution operator * is associative

$$\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} * \text{Image} = \begin{bmatrix}1\\2\\1\end{bmatrix} * \begin{bmatrix}1&2&1\end{bmatrix} * \text{Image} = \begin{bmatrix}1\\2\\1\end{bmatrix} * \left(\begin{bmatrix}1&2&1\end{bmatrix} * \text{Image} \right)$$

    - Complexity of filtering with nxn image and mxm kernel is O(nxnxmxm)
    - Complexity of filtering with nxn image and separable mxm kernel is O(2xnxnxm)
    - m/2 times faster implementation for Gaussian filtering!
- 3-sigma rule:
    - Pixels more than 3σ away can be ignored
    - No need for kernel size greater than (6σ+1)



Padding (why padding?)



Padding

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	2	3	4	5	0	0
0	0	6	7	8	9	10	0	0
0	0	11	12	13	14	15	0	0
0	0	16	17	18	19	20	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Constant (zero)

13	12	11	12	13	14	15	14	13
8	7	6	7	8	9	10	9	8
3	2	1	2	3	4	5	4	3
8	7	6	7	8	9	10	9	8
13	12	11	12	13	14	15	14	13
18	17	16	17	18	19	20	19	18
13	12	11	12	13	14	15	14	13
8	7	6	7	8	9	10	9	8

Mirror/Symmetric

1	1	1	2	3	4	5	5	5
1	1	1	2	3	4	5	5	5
1	1	1	2	3	4	5	5	5
6	6	6	7	8	9	10	10	10
11	11	11	12	13	14	15	15	15
16	16	16	17	18	19	20	20	20
16	16	16	17	18	19	20	20	20
16	16	16	17	18	19	20	20	20

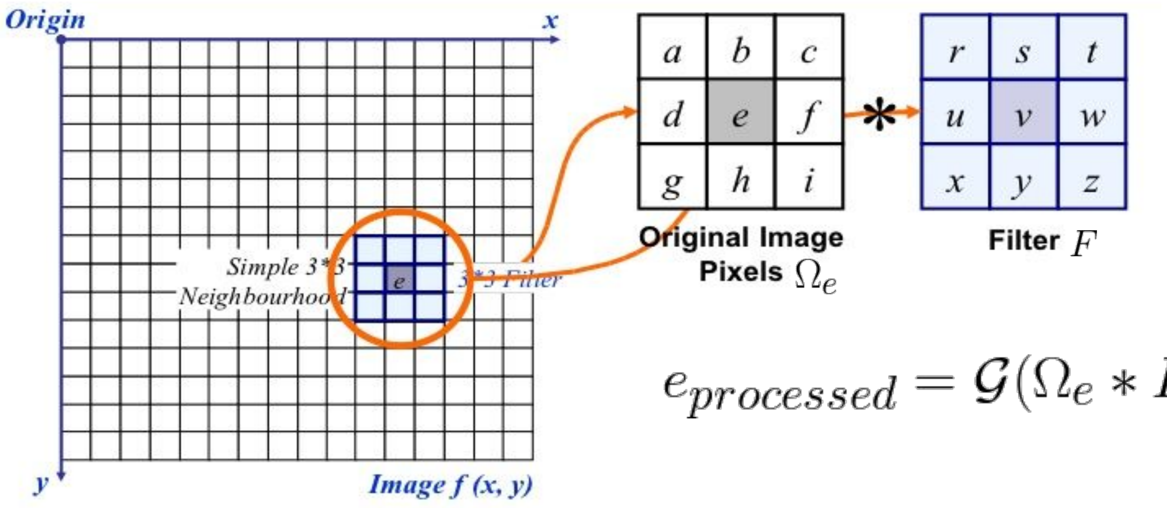
Replicate

Original Image

Filtered Image with Black Border

Filtered Image with Border Replication

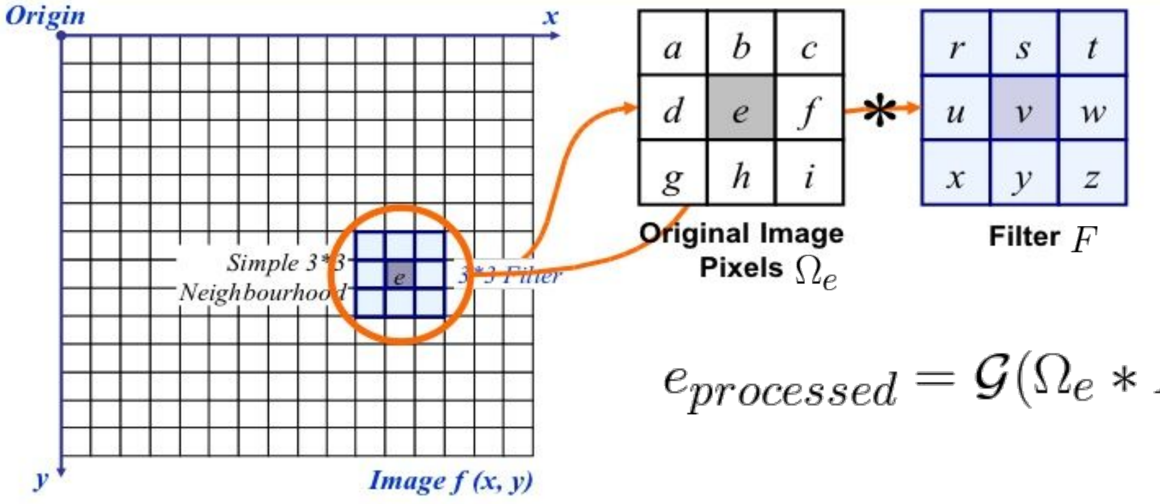
Morphological filter: Dilation



$$e_{processed} = \mathcal{G}(\Omega_e * F)$$

- Input is a binary image (True=1, False=0), padded with constant 0
- $*$ is the logical and
- $\mathcal{G}(\cdot)$ is the logical or reduction
- F is $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Morphological filter: Erosion



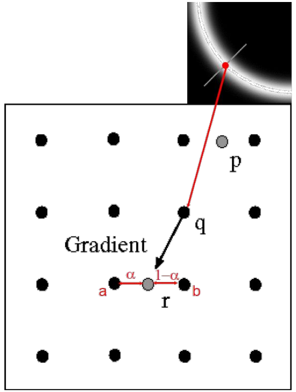
$$e_{processed} = \mathcal{G}(\Omega_e * F)$$

- Input is a binary image (True=1, False=0), padded with constant 1
- $*$ is the logical and
- $\mathcal{G}(\cdot)$ is the logical and reduction
- F is $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Q3 Edge detection algorithm (overview)

Key steps:

1. Pre-processing: smoothing e.g. Gauss filter (q.3b)
2. Compute intensity of gradients: e.g. Sobel filter, etc.
   a. With appropriate boundary handling (q.3a)
3. NMS along gradient direction (q.3c)
4. Returns boundary score



How to evaluate the performance?

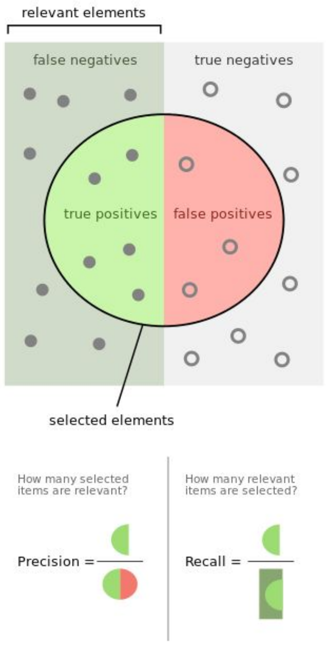
- We have a test set with ground truth binary labels for each pixel
  - True: is edge pixel
  - False: not edge pixel
- We have a model that outputs a score in [0,1]
  - How to make this model output binary decision? (bin_model = score + threshold)
  - How to select threshold?
- Edges are sparse: very few pixel are actually edge pixels
  - Measuring accuracy may not be the best idea...

Precision, recall and accuracy

- Precision: TP/(TP+FP)
- Recall: TP/(TP+FN)
- Accuracy: (TP+FN)/(TP+FP+TN+FN)

Suppose only 1% of pixels are true edge pixels in test sets, what are the precision, recall and accuracy of a model that randomly predicts edge score with th=0.99?

- Precision: 1%
- Recall : 1%
- Accuracy: 98%

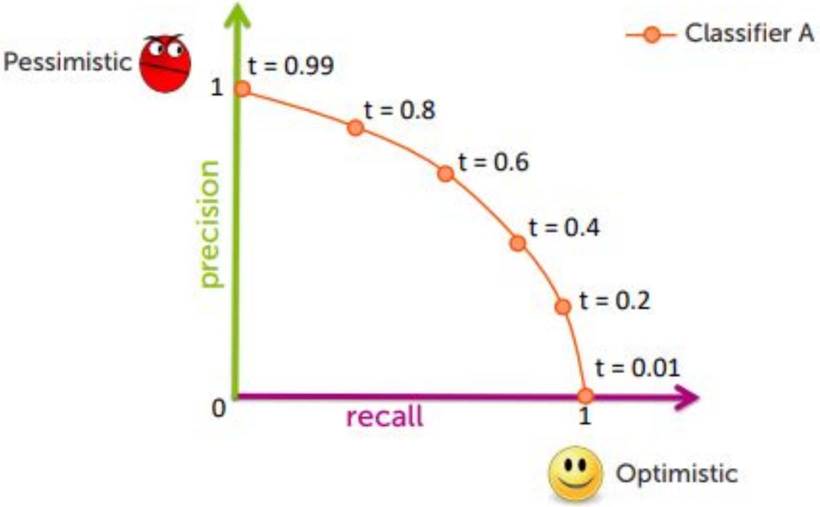


How are precision and recall related?

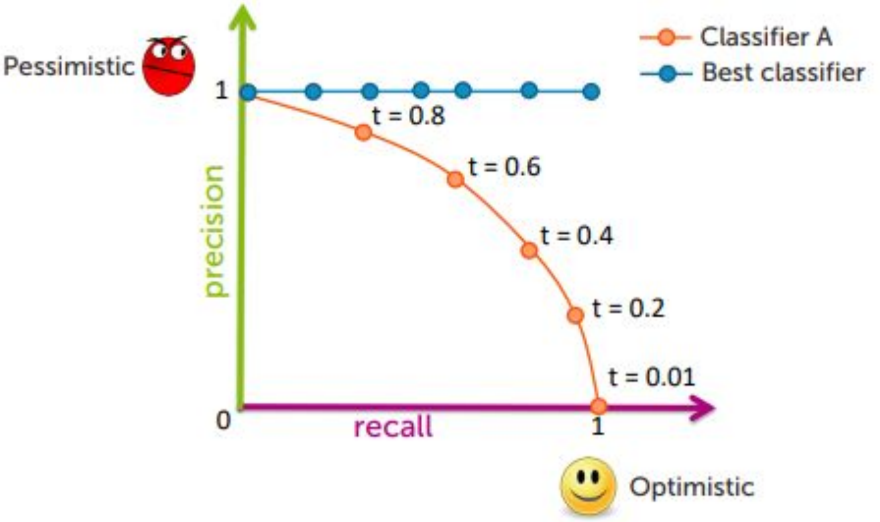
Generally, by increasing threshold

- Model gets more pessimistic (less likely to say yes)
- Precision likely increase
- Recall likely decrease

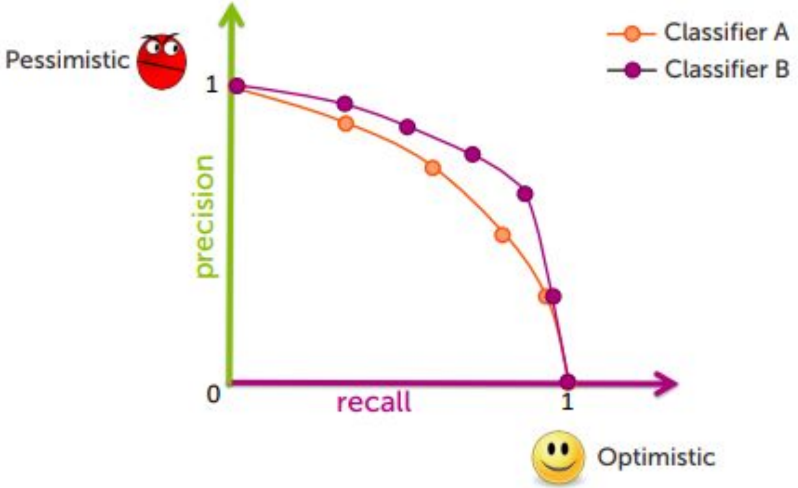
The precision-recall curve



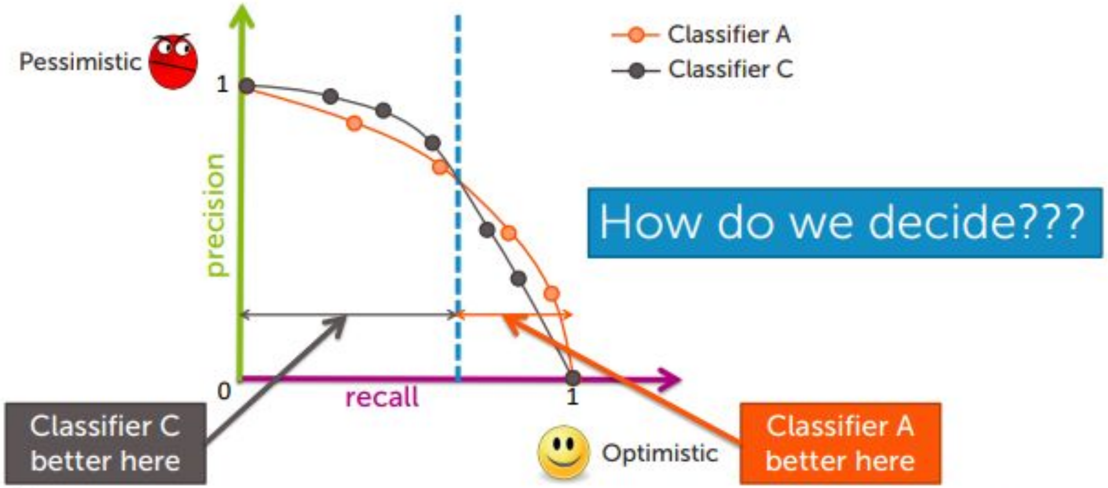
What does the perfect algorithm look like?



Which classifier is better? A or B?



Which classifier is better? A or C?



How to select the best predictor/threshold for a task?

Whichever maximizes F_β score:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

$\beta>1$ attaches greater importance to recall than to precision.

F_1 Score:

$$F_1 = \left(\frac{2}{\text{recall}^{-1} + \text{precision}^{-1}}\right) = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$