Binary Classification Using Fast Gaussian Filtering Algorithm

Kentaro Imajo, Keisuke Otaki,
Akihiro Yamamoto (Kyoto University)

Abstract

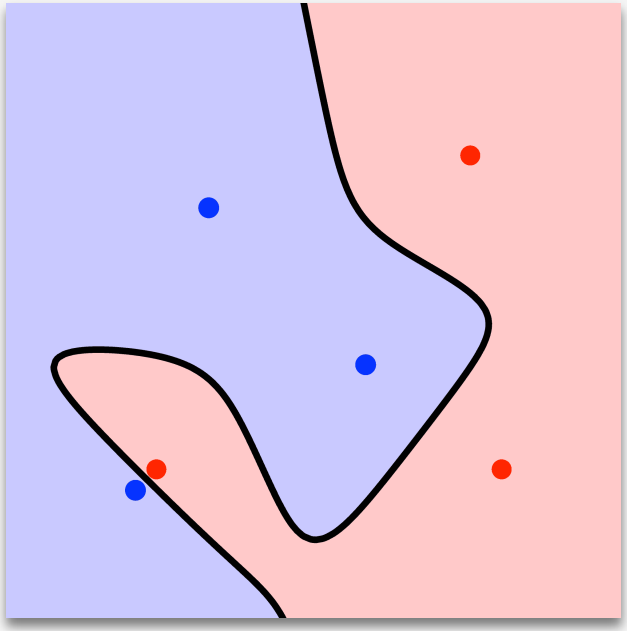
Speed up binary classification on 2D/3D
using the Gaussian kernel
faster than C-SVMs if # of data is 150,000+.

Training (pre-computation):

$$\mathrm{O}(n^3) \rightarrow \mathrm{O}(4^d n \log^{2d} n)$$

Prediction:

$$\mathrm{O}(n) \rightarrow \mathrm{O}(4^d \log^{2d} n)$$



Existing Work

C-SVM is a good binary classifier, but

- learning takes $\mathrm{O}(n^3)$ time,
- prediction takes $\mathrm{O}(n)$ time.

We cannot use C-SVMs if # of data gets large.

Solution

Training: $\mathrm{O}(n^3) \rightarrow$ We don't learn.

But it takes $\mathbf{O}(4^d n \log^{2d} n)$ for pre-computation.

Treat every datum equally.

- For future works,
  we want to pretend learning of C-SVM, though.

Prediction: $\mathrm{O}(n) \rightarrow \mathrm{O}(4^d \log^{2d} n)$

Speed it up
using our fast Gaussian filtering algorithm.

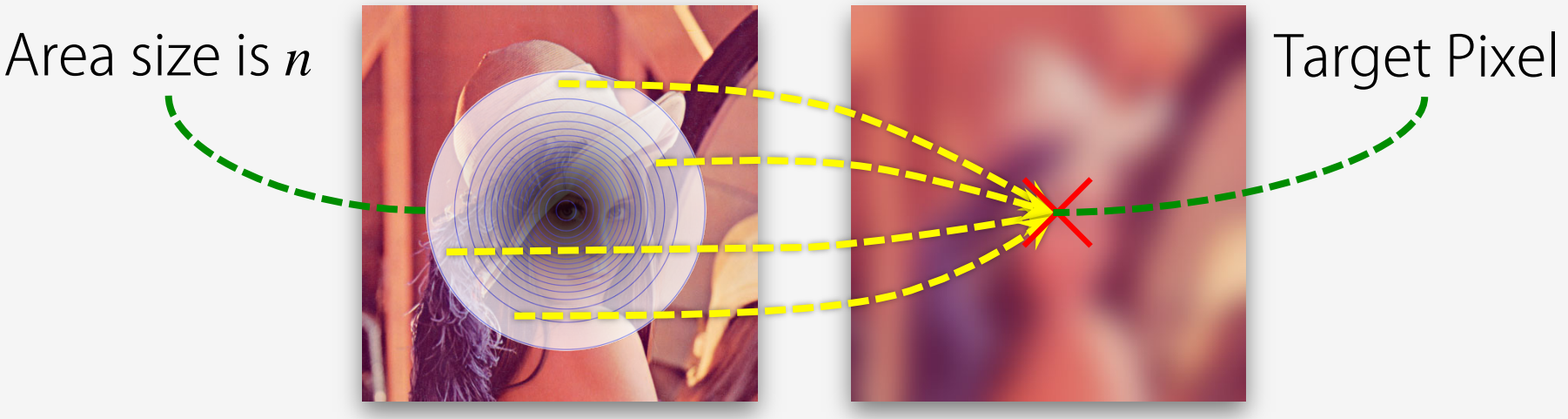
Outline

1. Fast Gaussian filtering algorithm (FGFA)
   - A fundamental algorithm of this study
2. Binary classification using FGFA
   - Extension of the previous work for binary classification
3. Evaluations
4. Conclusions

1. Fast Gaussian Filtering Algorithm
2. Binary Classification Using
   The Fast Gaussian Filtering Algorithm
3. Evaluations
4. Conclusions

Benefit of FGFA

The fast Gaussian filtering algorithm computes one Gaussian-filtered pixel in constant time to the area size n ($\propto \sigma^2$).



The naïve method requires $O(n)$ time.

Key Ideas of FGFA

- The Gaussian function can be approximated with a spline.
- Splines become discrete if they are differentiated some times.
- Convolution of an image and a spline would be computed fast since convolution of an image and a discrete function is easy.

Gaussian Function

The Gaussian filter is computed by convolutions with the 2D Gaussian function.

The 2D Gaussian Function:

Circle

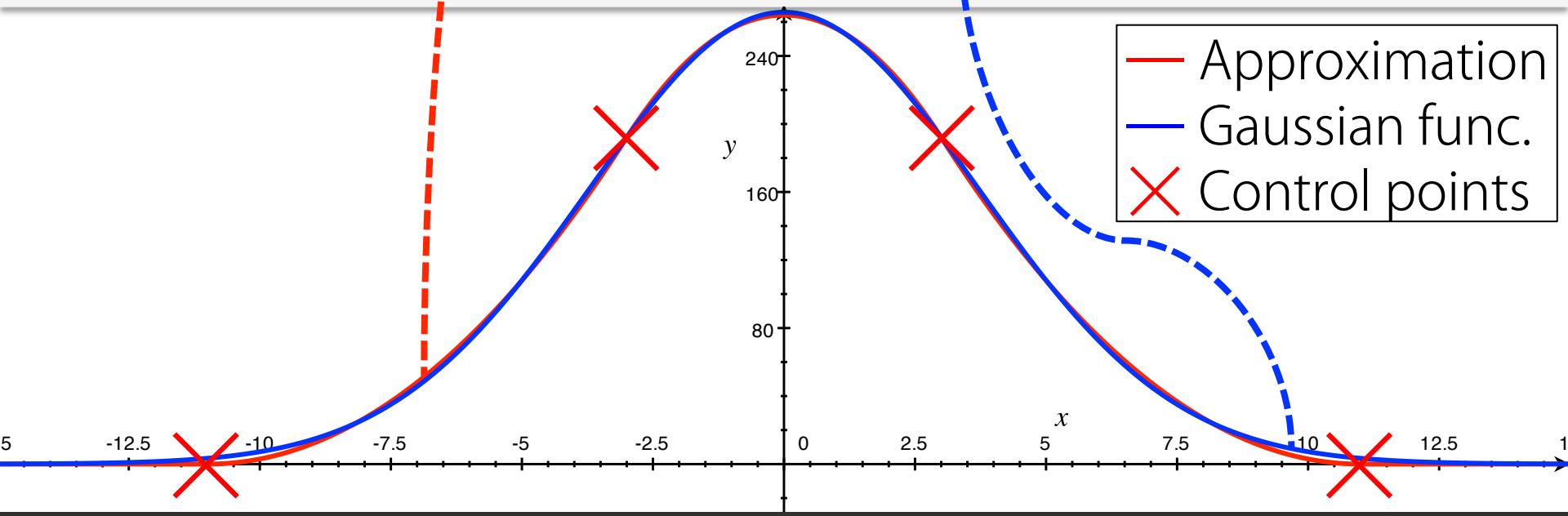
$$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

where a parameter σ is variance.

Approximation

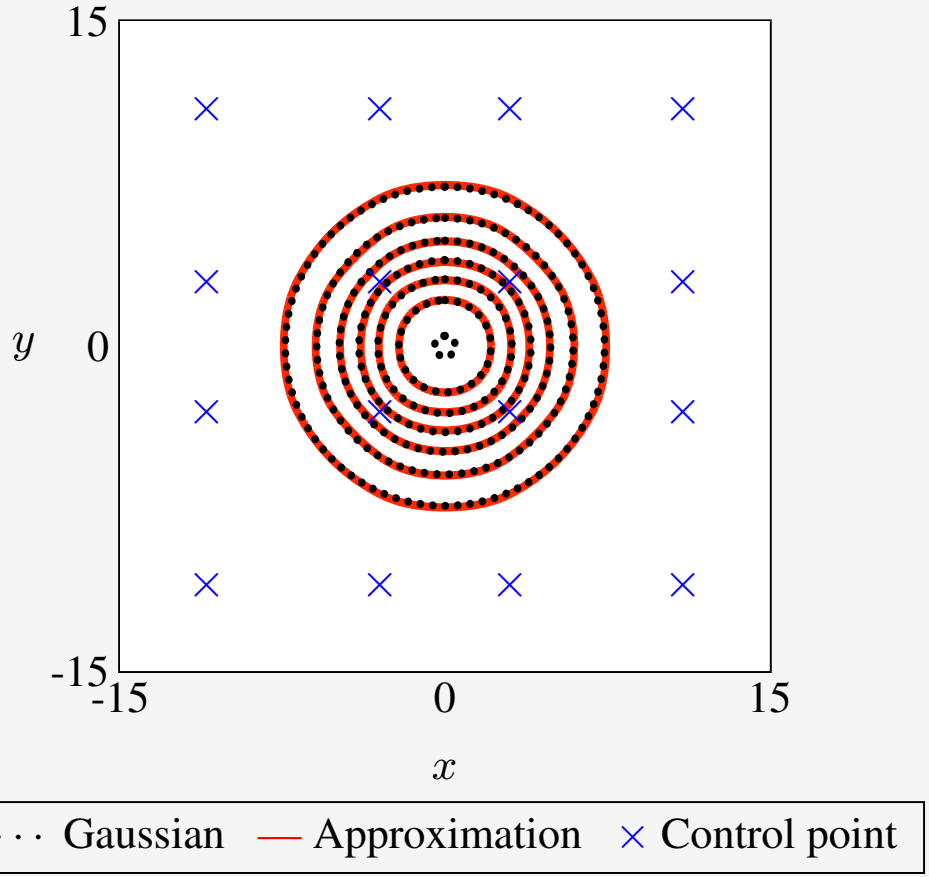
$$\tilde{\psi}(x) = 3(x+11)_+^2 - 11(x+3)_+^2 + 11(x-3)_+^2 - 3(x-11)_+^2,$$

$$\simeq 2.657 \times 10^2 \exp\left(-\frac{x^2}{5.2720^2}\right).$$



Approximation of 2D

$$\tilde{\psi}(x, y) = \tilde{\psi}(x)\tilde{\psi}(y)$$

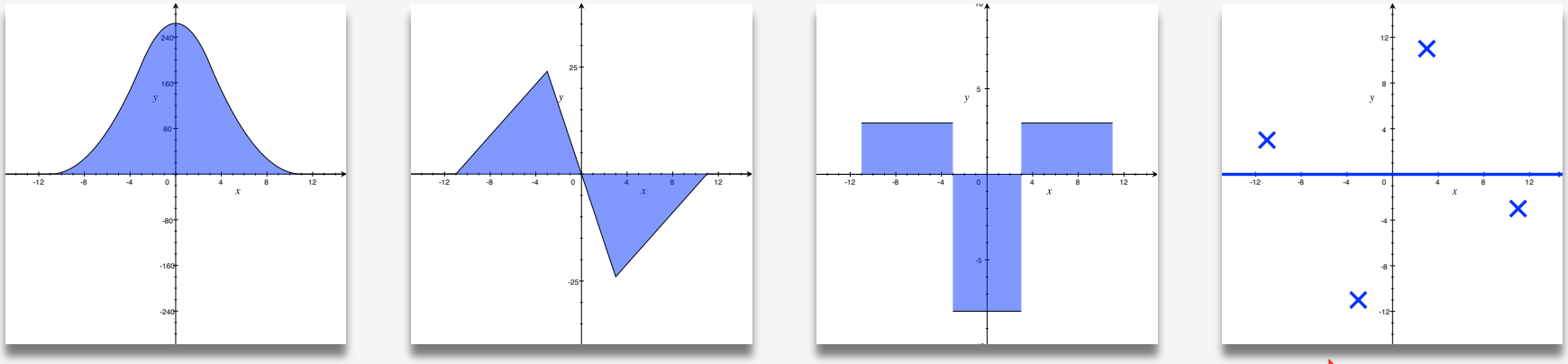


Properties of the Function

- It is a 2nd order spline.
  - Piecewise polynomial function
- It has only 2% approximation error
  - $(\int |\tilde{\psi}(x) - \psi(x)|dx) \ / \ (\int |\psi(x)|dx)$
- It has only 4 control points
- It has unequal intervals of control points while general approximation functions often have equal intervals.
  - It is sufficiently optimized as a spline.

Benefit of Splines

Differentiating splines, they get constant.



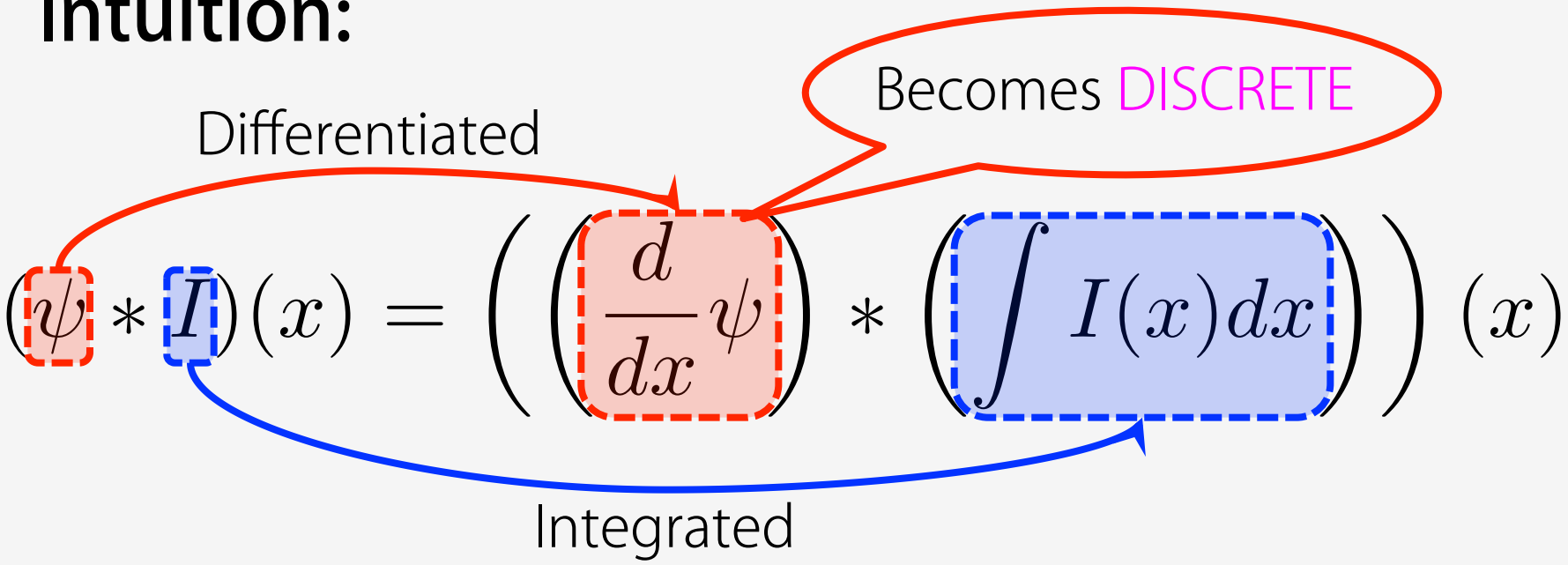
Differentiate 3-times

The Gaussian-like function
can also become discrete!

Speed Up of Convolution

Convolution can be reduced by differentiating splines until they become discrete.

Intuition:



Transformation

Transform convolution into summation

$$
\begin{aligned}
(\tilde{\psi} * I)(x) &= \sum_{\Delta x \in \mathbb{Z}} \tilde{\psi}(\Delta x) I(x - \Delta x) \\
&= \sum_{\Delta x \in \mathbb{Z}} \left(\sum_{i=0}^{m} a_i (\Delta x - b_i)_+^n \right) I(x - \Delta x) \\
&= \sum_{i=0}^{m} a_i J(x - b_i), \quad J(x) = \sum_{\Delta x \in \mathbb{Z}} \Delta x_+^n I(x - \Delta x)
\end{aligned}
$$

$O(m)$ linear combination Pre-computed

2D Convolution

The 2D Gaussian function is decomposed:

$$\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) = \exp\left(-\frac{x^2}{2\sigma^2}\right)\exp\left(-\frac{y^2}{2\sigma^2}\right)$$

The approximation can be defined as:

$$(\tilde{\psi} * I)(x,y) = \sum_{i=1}^{m} a_i \sum_{j=1}^{m} a_j J(x-b_i, y-b_j)$$

where $J(x,y) = \sum_{(\Delta x, \Delta y)\in\mathbb{Z}_+^2} \Delta x^n \Delta y^n I(x-\Delta x, y-\Delta y)$

$O(m^2)$ linear combination

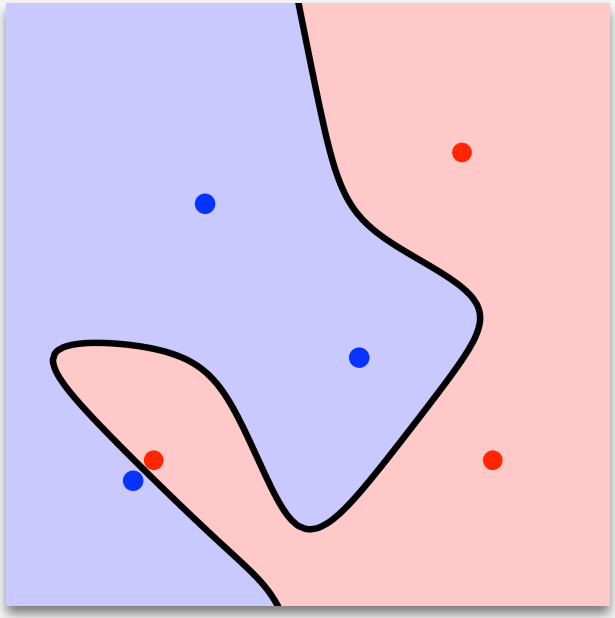
Overview of FGFA

Using $J(x,y)$, which is an integrated image of an input image, we can compute every pixel apart in constant time to the area size.

1. Fast Gaussian Filtering Algorithm
2. Binary Classification Using The Fast Gaussian Filtering Algorithm
3. Evaluations
4. Conclusions

Naïve FGFA

If we regard 2D/3D spaces as images,
we can determine the class of any point
in constant time using the naïve FGFA.

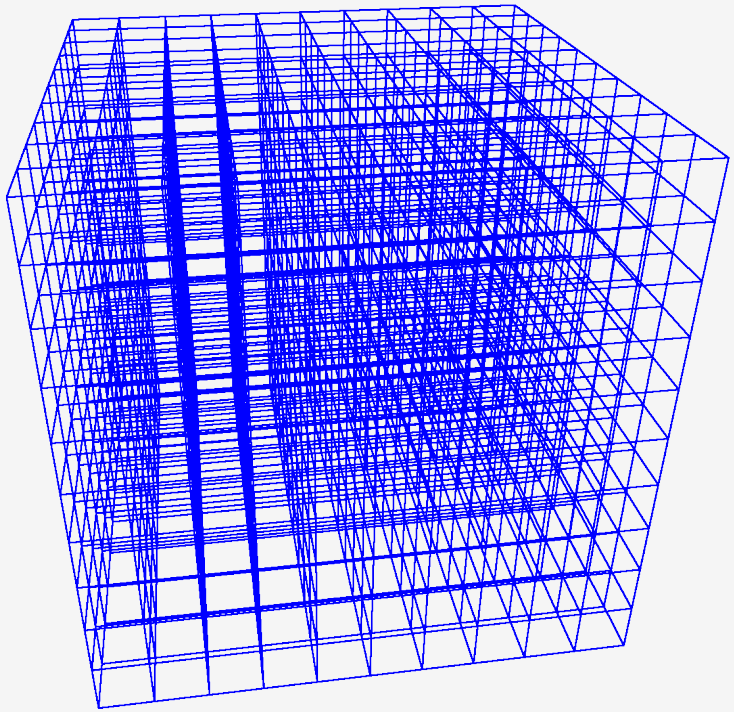


Problem

Data of images (especially 3D images) exhaust memory resources.

If the size of each coordinate is 1000, # of blocks gets 10^9.

This is often lager than # of training data.



Goal

Extend FGFA to apply to sparse images.

However, FGFA computes this expression:

$$(\tilde{\psi} * I)(x, y) = \sum_{i=1}^{m} a_i \sum_{j=1}^{m} a_j J(x - b_i, y - b_j)$$

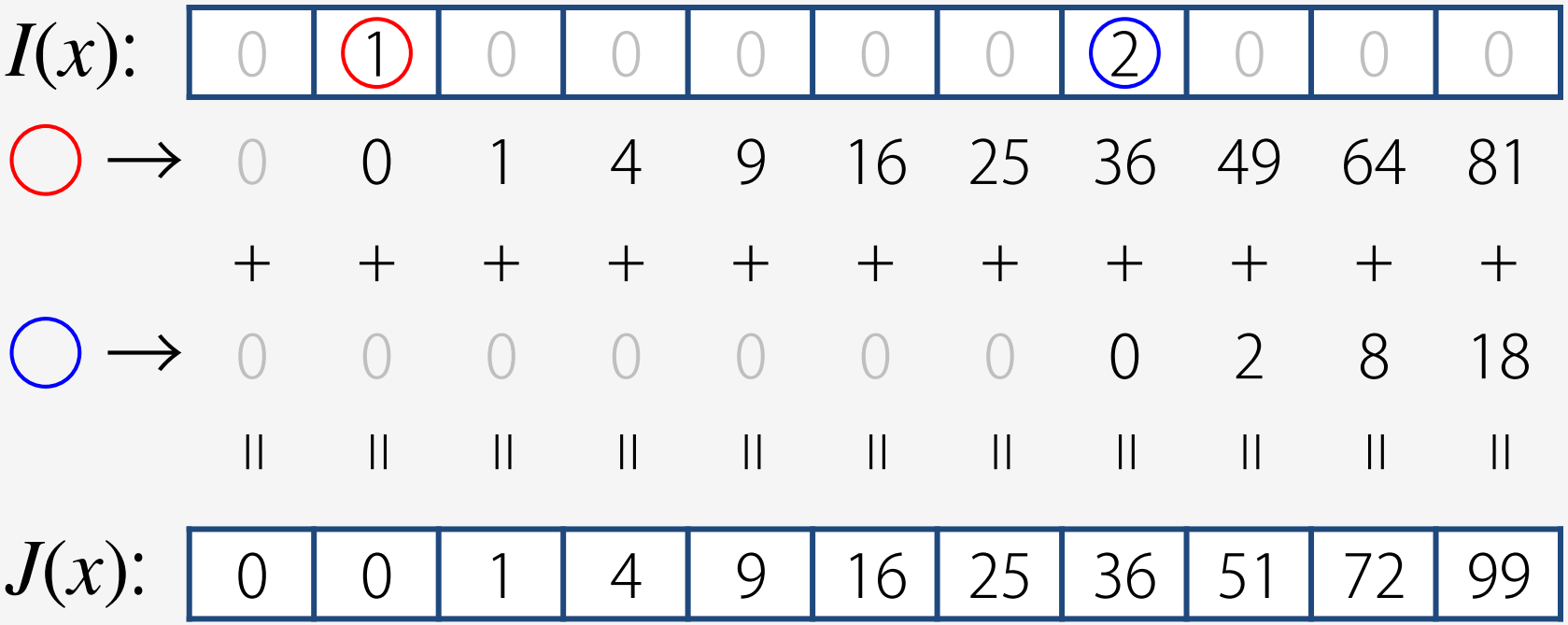
$$J(x, y) = \sum_{(\Delta x, \Delta y) \in \mathbb{Z}_+^2} \Delta x^n \Delta y^n I(x - \Delta x, y - \Delta y)$$

Badly, $J(x,y)$ cannot be pre-computed because of memory space.

What Is J(x)?

$J(x)$ is an integrated array of an input $I(x)$.

1D Case:

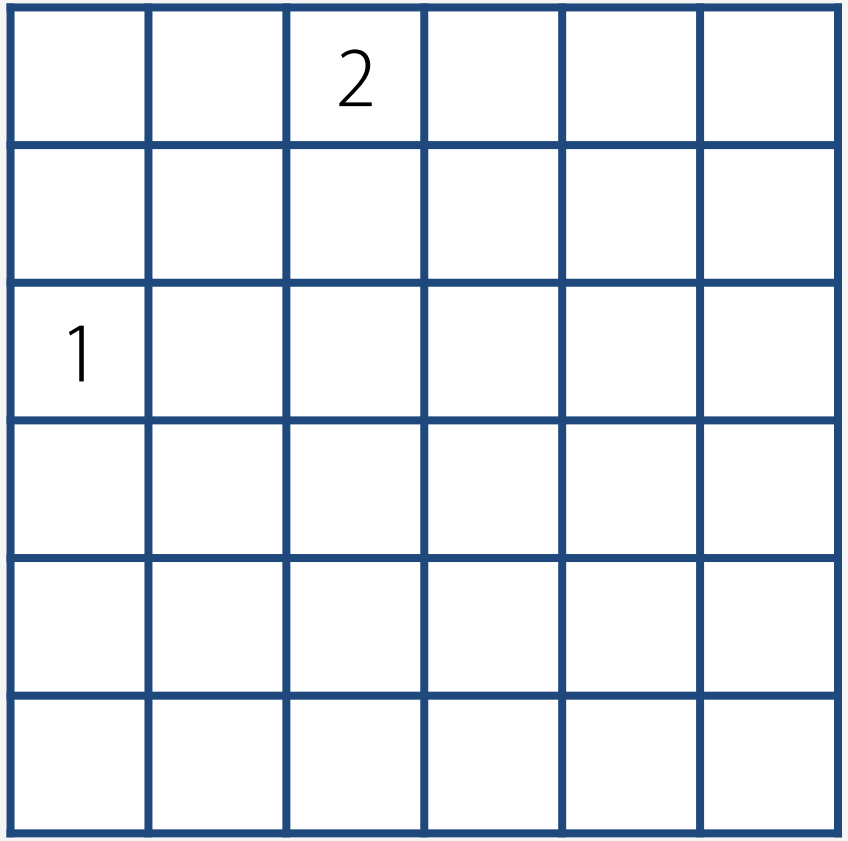


What Is J(x,y)?

$J(x,y)$ is an extend version of $J(x)$ for 2D.

If such $I(x,y)$ is given,

$J(x,y)$ gets...

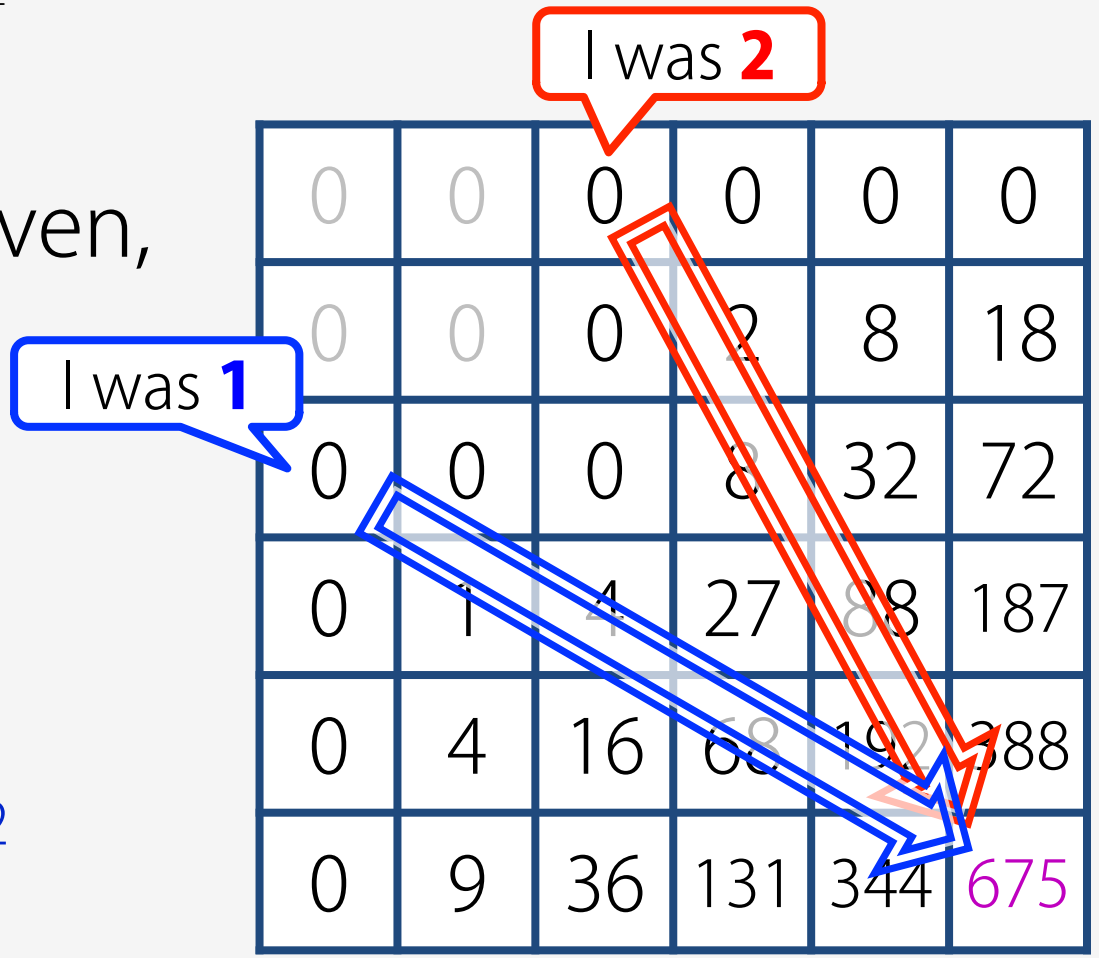


What Is J(x,y)?

$$J(x,y) = \sum_{(\Delta x, \Delta y) \in \mathbb{Z}_+^2} \Delta x^n \Delta y^n I(x - \Delta x, y - \Delta y)$$

If such $I(x,y)$ is given,

$J(x,y)$ gets this.



Example:

675 =

$2 \cdot 3^2 \cdot 5^2 + 1 \cdot 5^2 \cdot 3^2$

Solution

The proposed method builds a data structure that can answer $J(x,y)$ fast using range trees.

Task:

Answer $J(x,y)$ fast, but you cannot have the size of combinations of coordinate values.

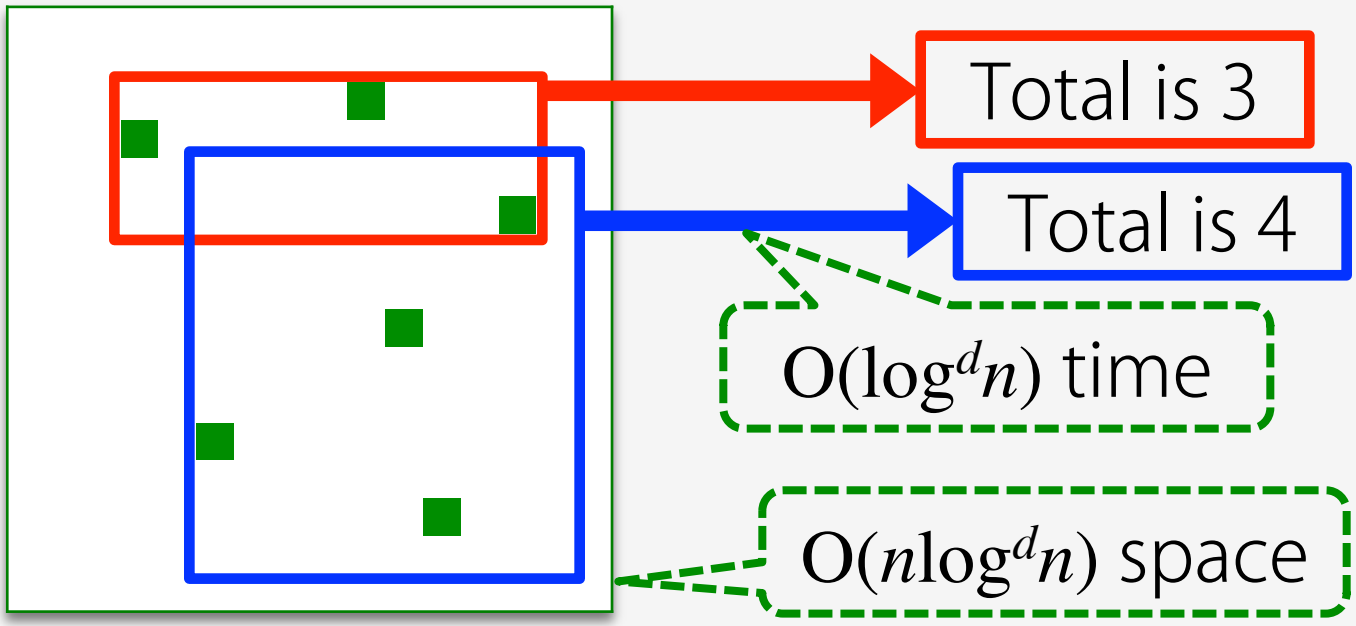
$$J(x,y) = \sum_{(\Delta x, \Delta y) \in \mathbb{Z}_+^2} \Delta x^n \Delta y^n I(x - \Delta x, y - \Delta y)$$

Many combinations

Sparse

Range Tree

A d-dimensional range tree can compute the total sum in a box $\mathbf{x}_1$-$\mathbf{x}_2$ in $O(\log^d n)$ time with $O(n\log^d n)$ space.



Deformation of J(x)

Range tree can give $\sum_{i=a,\cdots,b} f(x)I(x)$,

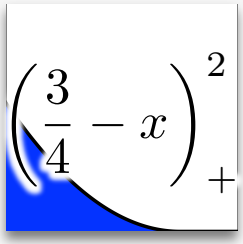
but J is $\sum_{i=0,\cdots,a} (a-x)^n I(x)$, which contains a.

It can be deformed into:

$$\sum_{i=0,\cdots,a} (a-x)^2 I(x) = \sum_{i=0,\cdots,a} x^2 I(x) - 2a \sum_{i=0,\cdots,a} xI(x) + a^2 \sum_{i=0,\cdots,a} I(x)$$

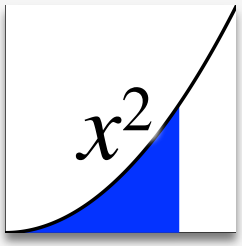
Deformation of J(x)

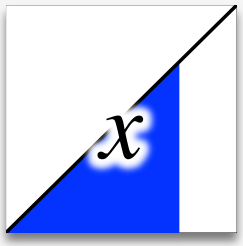
Example (1D case):

If you want to calculate $J(3/4)$ 

,

you can calculate it using three K:



-3/2


+9/16


.

Deformation of J(x,y)

Range tree can give:

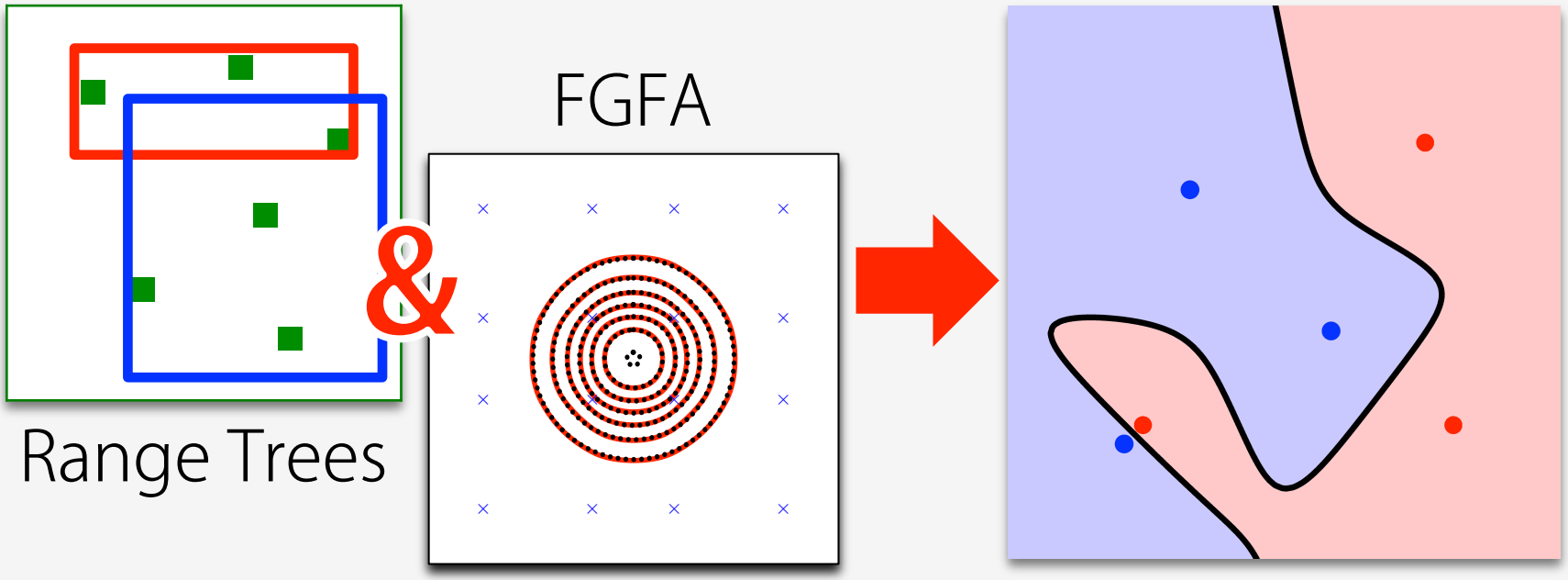
$$K_{i,j}(x,y) = \sum_{(\Delta x, \Delta y) \in \{(0,0),\cdots,(x,y)\}} \Delta x^i \, \Delta y^j \, I(\Delta x, \Delta y).$$

Using this, J can be computed with:

$$J(x,y) = \begin{pmatrix} y^2 \\ -2y \\ 1 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} K_{0,0}(x,y) & K_{1,0}(x,y) & K_{2,0}(x,y) \\ K_{0,1}(x,y) & K_{1,1}(x,y) & K_{2,1}(x,y) \\ K_{0,2}(x,y) & K_{1,2}(x,y) & K_{2,2}(x,y) \end{pmatrix} \begin{pmatrix} x^2 \\ -2x \\ 1 \end{pmatrix}.$$

Binary Classifier

The proposed binary classifier can decide the class of any point in $O(4^d \log^{2d} n)$ time, and the pre-computation is $O(4^d n \log^{2d} n)$ time.



1. Fast Gaussian Filtering Algorithm
2. Binary Classification Using
   The Fast Gaussian Filtering Algorithm
3. Evaluations
4. Conclusions

Relations to C-SVM

Objective function of C-SVM:

$$\max: \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \exp\left(-\frac{|x_i - x_j|^2}{\sigma^2} \right),$$

$$\text{s.t.}: \quad 0 \leq \alpha_i \leq C \ (i = 1, \cdots, n), \ \sum_{i=1}^{n} \alpha_i y_i = 0,$$

Proposed method:

$$\max: \quad \sum_{i=1}^{n} \alpha_i,$$

$$\text{s.t.}: \quad 0 \leq \alpha_i \leq C \ (i = 1, \cdots, n).$$

– The solution is $\alpha_i = C$.

Experiments for Accuracy

Accuracy of binary classification:

	Proposed Method	C-SVM			
		$C = 0.1$	$C = 1$	$C = 10$	$C = 10^6$
$\sigma = 0.1$	93%	58%	68%	69%	69%
$\sigma = 0.3$	93%	83%	94%	93%	93%
$\sigma = 1$	93%	93%	95%	93%	92%
$\sigma = 3$	86%	90%	94%	95%	89%
$\sigma = 10$	84%	85%	90%	93%	92%

Figure Accuracy for Iris Flower Data Set

It is not so worse than C-SVM.

Experiments for Time

CPU time of training and prediction:

# of training data		1,000	10,000	100,000	1,000,000
Proposed method	Prediction (Precomputation)	2.43 ms (0.13 s)	2.48 ms (1.48 s)	2.55 ms (17.8 s)	2.96 ms (203 s)
LIBSVM	Prediction (Training)	0.02 ms (0.05 s)	0.22 ms (4.47 s)	2.29 ms (442 s)	23.6 ms (1000+ s)

Figure Time of Training and Prediction for 2D Data

The proposed method is faster
when # of training data is over 150,000.

1. Fast Gaussian Filtering Algorithm
2. Binary Classification Using
   The Fast Gaussian Filtering Algorithm
3. Evaluations
4. Conclusions

Conclusions

We proposed a new binary classifier.

- It is not so worse than C-SVM, and it can also speed up prediction of C-SVM.
- Pre-computation is $\mathrm{O}(4^d n \log^{2d} n)$ time, and prediction is $\mathrm{O}(4^d \log^{2d} n)$ time.

The proposed method consists of

- Fast Gaussian filtering algorithm,
- Range trees.