IT4I

Machine learning and fuzzy transform

Pavel Vlašánek

Institute for Research and Applications of Fuzzy Modeling
University of Ostrava

pavel.vlasanek@osu.cz

Seminář strojového učení a modelování

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007-13 OP Výzkum a vývoj pro inovace

VISIONIRAFM

IRAFM

Decision making

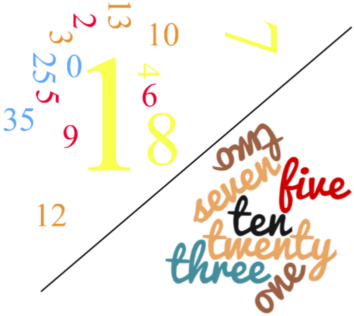
- Is this a person?
- Is this a number?
- Is this a ...

YES/NO

Decision making

- Is this a person?
- Is this a number?
- Is this a ...

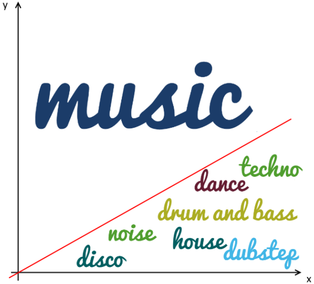
YES/NO



Decision making

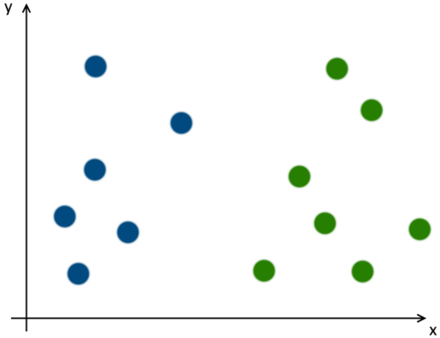
- Is this a person?
- Is this a number?
- Is this a ...

YES/NO



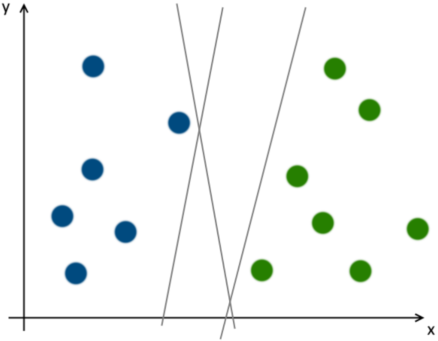
Binary classification

- Two types of linearly separable data in the set.
- What is the ideal separation line?
- What to do if data are not linearly separable?



Binary classification

- Two types of linearly separable data in the set.
- What is the ideal separation line?
- What to do if data are not linearly separable?



Binary classification

- Two types of linearly separable data in the set.
- What is the ideal separation line?
- What to do if data are not linearly separable?

Support vector machine

Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

Support vector machine

- Data are separated by *hyperplane* with dimension one less than number of feature parameters.
  - For our example of 2D points, the separation will be given by a line.
- The line is defined such as $w^T x + b = y$.
- Let's draw two lines, as close to the two groups as possible.
  - The points closest to them are called *support vector*.

Support vector machine

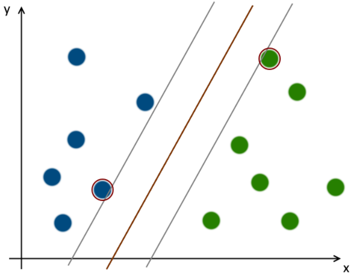
- Data are separated by *hyperplane* with dimension one less than number of feature parameters.
  - For our example of 2D points, the separation will be given by a line.
- The line is defined such as $w^T x + b = y$.
- Let's draw two lines, as close to the two groups as possible.
  - The points closest to them are called *support vector*.

Support vector machine

- Data are separated by *hyperplane* with dimension one less than number of feature parameters.
  - For our example of 2D points, the separation will be given by a line.
- The line is defined such as $w^T x + b = y$.
- Let's draw two lines, as close to the two groups as possible.
  - The points closest to them are called *support vector*.

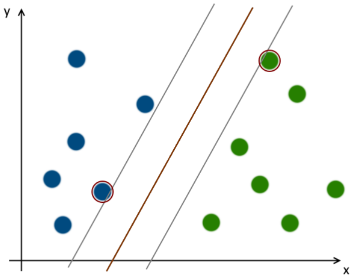
Support vector machine

- Data are separated by *hyperplane* with dimension one less than number of feature parameters.
  - For our example of 2D points, the separation will be given by a line.
- The line is defined such as $w^T x + b = y$.
- Let's draw two lines, as close to the two groups as possible.
  - The points closest to them are called *support vector*.



Support vector machine

- Data are separated by *hyperplane* with dimension one less than number of feature parameters.
  - For our example of 2D points, the separation will be given by a line.
- The line is defined such as $w^T x + b = y$.
- Let's draw two lines, as close to the two groups as possible.
  - The points closest to them are called *support vector*.



Support vector machine

- Data are separated by *hyperplane* with dimension one less than number of feature parameters.
  - For our example of 2D points, the separation will be given by a line.
- The line is defined such as $w^T x + b = y$.
- Let's draw two lines, as close to the two groups as possible.
  - The points closest to them are called *support vector*.

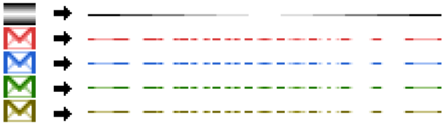
Let hyperplane (line) closer to one group is given such as $w^T x + b = -1$ and second one such as $w^T x + b = +1$. Our target is to maximise the distance between them.

IRAFM

Feature vector

For a simplification of our data, we should use another representation.

- Colours.
- F-transform components.
- Gradients.



Feature vector

For a simplification of our data, we should use another representation.

- Colours.
- F-transform components.
- Gradients.

Feature vector

For a simplification of our data, we should use another representation.

- Colours.
- F-transform components.
- Gradients.

Vlašánek, P. and Perfilieva, I. "Patch based inpainting inspired by the F^1-transform". International Journal of Hybrid Intelligent Systems. 2016, č. 13, s. 39-48. ISSN 1448-5869.

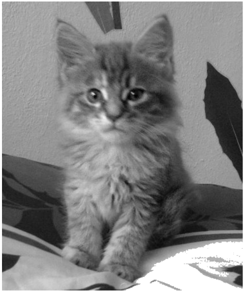
Hurtík, P., Hodáková, P. and Perfilieva, I. "Approximate Pattern Matching Algorithm." International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. Springer International Publishing, 2016.

IRAFM

Feature vector

For a simplification of our data, we should use another representation.

- Colours.
- F-transform components.
- Gradients.

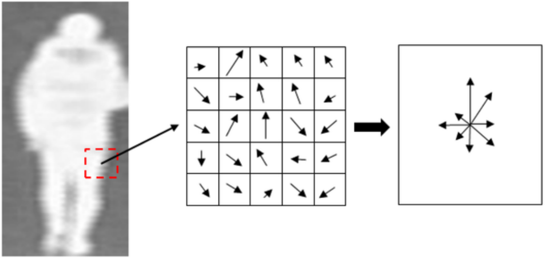


```
165, 165, 166, 166, 162, 158, 161, 127,  95, 100, 158, 160, 148, 145, 159, 158, 158, 159, 161;
164, 165, 166, 166, 154, 137, 151, 152, 114, 114, 157, 151, 117, 116, 157, 158, 158, 160, 161;
164, 165, 165, 165, 148, 121, 130, 141, 139, 137, 138, 127, 105, 110, 155, 159, 159, 161, 160;
164, 164, 164, 163, 149, 112, 120, 119,  99,  91,  96, 106, 103, 113, 156, 159, 157, 161, 160;
163, 163, 164, 163, 154, 124, 119, 108,  85,  71,  67,  77,  92, 109, 153, 159, 121, 142, 159;
161, 162, 162, 162, 152, 133, 119, 116, 118,  74,  72,  67,  76,  99, 152, 157,  87,  64, 135;
159, 160, 162, 161, 155, 142, 125, 107, 136,  99,  65,  67,  61,  87, 147, 154,  74,  39,  76;
158, 158, 159, 159, 156, 141, 124, 120, 131, 105,  72,  66,  51,  66, 136, 152,  73,  39,  45;
154, 155, 155, 156, 156, 156, 142, 159, 161, 106,  85,  79,  69,  87, 135, 154,  96,  39,  38;
151, 152, 153, 153, 152, 166, 173, 183, 174, 137, 113,  96, 101, 109, 139, 155, 143,  57,  36;
148, 149, 151, 151, 149, 145, 152, 167, 171, 151, 111,  87,  86,  90, 128, 154, 156,  91,  36;
108, 119, 131, 140, 144, 147, 153, 165, 176, 160, 133, 113,  93,  84, 112, 152, 157, 135,  56;
 68,  78,  85,  91, 102, 137, 164, 169, 169, 153, 136, 124,  95,  66,  89, 144, 155, 153, 109;
 48,  62,  71,  75,  84, 131, 171, 172, 157, 133, 110,  98,  71,  46,  61, 109, 125, 131, 133;
 43,  44,  50,  51,  66, 130, 170, 172, 153, 131, 105,  84,  52,  42,  53,  62,  62,  65,  74;
 57,  52,  42,  39,  50, 110, 156, 152, 149, 121, 111,  84,  47,  57,  53,  42,  46,  50,  52;
 82,  66,  39,  35,  47,  98, 142, 133, 106, 121, 122,  91,  51,  47,  44,  31,  43,  55,  54;
110,  99,  74,  47,  40,  96, 146, 124,  75, 143, 128,  85,  44,  40,  45,  57,  82,  98, 102;
150, 145, 128, 101,  83, 128, 137, 116, 111, 151, 113,  77,  49,  79, 114, 142, 161, 173, 176;
 68, 108, 166, 160, 148, 151, 113,  92, 134, 125, 103,  97, 139, 176, 188, 185, 184, 182, 176;
120, 158, 177, 179, 179, 167, 151, 135, 138, 114,  98, 137, 199, 216, 215, 201, 185, 170, 118;
179, 180, 191, 197, 197, 185, 172, 146, 109, 128, 143, 186, 201, 205, 219, 228, 221, 201, 150]
```

Feature vector

For a simplification of our data, we should use another representation.

- Colours.
- F-transform components.
- Gradients.



Histogram of Oriented gradients

Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). Vol. 1. IEEE, 2005.

Histogram of Oriented gradients

- Gradient computation.
- The HoG method is used in a *Scale-invariant feature transform* (SIFT).
- Variant proposed by Dalal and Navneet.
  - Each cell is 8×8 pixels big.
  - Four cells are connected to 16×16 block.
  - For proposed size 64×128 pixels for a person, we receive 105 blocks in total.

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Histogram of Oriented gradients

- Gradient computation.
- The HoG method is used in a *Scale-invariant feature transform* (SIFT).
- Variant proposed by Dalal and Navneet.
  - Each cell is 8×8 pixels big.
  - Four cells are connected to 16×16 block.
  - For proposed size 64×128 pixels for a person, we receive 105 blocks in total.

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$g_x = [-1, 0, 1]$$

$$g_y = [-1, 0, 1]^T$$

IRAFM

Histogram of Oriented gradients

- Gradient computation.
- The HoG method is used in a *Scale-invariant feature transform* (SIFT).
- Variant proposed by Dalal and Navneet.
  - Each cell is 8×8 pixels big.
  - Four cells are connected to 16×16 block.
  - For proposed size 64×128 pixels for a person, we receive 105 blocks in total.

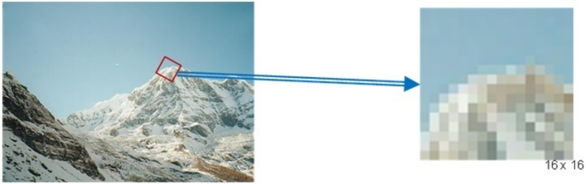
$$\theta = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

$$|g_x, g_y| = \sqrt{g_x^2 + g_y^2}$$

IRAFM

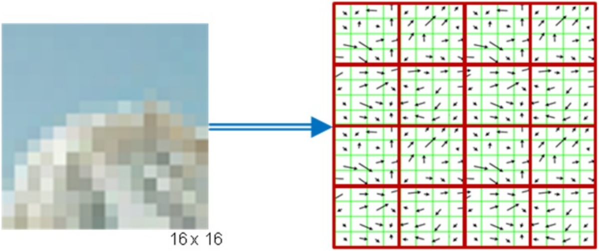
Histogram of Oriented gradients

- Gradient computation.
- The HoG method is used in a *Scale-invariant feature transform* (SIFT).
- Variant proposed by Dalal and Navneet.
  - Each cell is 8×8 pixels big.
  - Four cells are connected to 16×16 block.
  - For proposed size 64×128 pixels for a person, we receive 105 blocks in total.



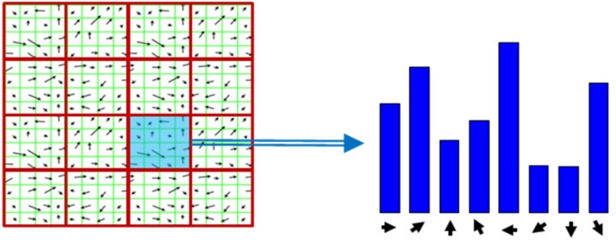
Histogram of Oriented gradients

- Gradient computation.
- The HoG method is used in a *Scale-invariant feature transform* (SIFT).
- Variant proposed by Dalal and Navneet.
  - Each cell is 8×8 pixels big.
  - Four cells are connected to 16×16 block.
  - For proposed size 64×128 pixels for a person, we receive 105 blocks in total.



Histogram of Oriented gradients

- Gradient computation.
- The HoG method is used in a *Scale-invariant feature transform* (SIFT).
- Variant proposed by Dalal and Navneet.
  - Each cell is 8×8 pixels big.
  - Four cells are connected to 16×16 block.
  - For proposed size 64×128 pixels for a person, we receive 105 blocks in total.

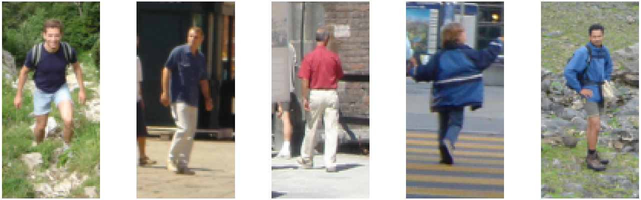


Histogram of Oriented gradients

- Gradient computation.
- The HoG method is used in a *Scale-invariant feature transform* (SIFT).
- Variant proposed by Dalal and Navneet.
  - Each cell is 8×8 pixels big.
  - Four cells are connected to 16×16 block.
  - For proposed size 64×128 pixels for a person, we receive 105 blocks in total.

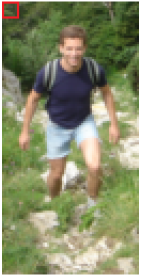
Histogram of Oriented gradients

- Gradient computation.
- The HoG method is used in a *Scale-invariant feature transform* (SIFT).
- Variant proposed by Dalal and Navneet.
  - Each cell is 8×8 pixels big.
  - Four cells are connected to 16×16 block.
  - For proposed size 64×128 pixels for a person, we receive 105 blocks in total.



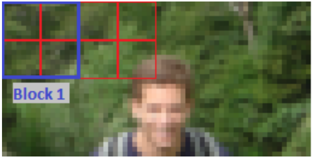
Histogram of Oriented gradients

- Gradient computation.
- The HoG method is used in a *Scale-invariant feature transform* (SIFT).
- Variant proposed by Dalal and Navneet.
  - Each cell is 8×8 pixels big.
  - Four cells are connected to 16×16 block.
  - For proposed size 64×128 pixels for a person, we receive 105 blocks in total.



Histogram of Oriented gradients

- Gradient computation.
- The HoG method is used in a *Scale-invariant feature transform* (SIFT).
- Variant proposed by Dalal and Navneet.
  - Each cell is 8×8 pixels big.
  - Four cells are connected to 16×16 block.
  - For proposed size 64×128 pixels for a person, we receive 105 blocks in total.

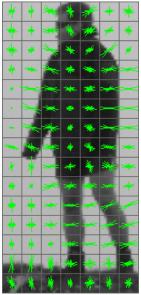


Histogram of Oriented gradients

- Gradient computation.
- The HoG method is used in a *Scale-invariant feature transform* (SIFT).
- Variant proposed by Dalal and Navneet.
  - Each cell is 8×8 pixels big.
  - Four cells are connected to 16×16 block.
  - For proposed size 64×128 pixels for a person, we receive 105 blocks in total.

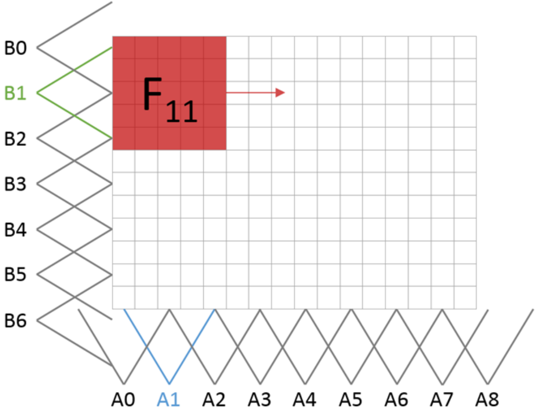
Histogram of Oriented gradients

- Gradient computation.
- The HoG method is used in a *Scale-invariant feature transform* (SIFT).
- Variant proposed by Dalal and Navneet.
  - Each cell is 8×8 pixels big.
  - Four cells are connected to 16×16 block.
  - For proposed size 64×128 pixels for a person, we receive 105 blocks in total.



F-transform

- The separate regions are processed independently.
- Kernel is based on the basic functions.



F-transform

- The separate regions are processed independently.
- Kernel is based on the basic functions.

$$g = AB^T$$

B

$$\begin{pmatrix} 0 \\ .5 \\ 1 \\ .5 \\ 0 \end{pmatrix} \boxed{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & .25 & .5 & .25 & 0 \\ 0 & .5 & 1 & .5 & 0 \\ 0 & .25 & .5 & .25 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}}$$

$$(0, .5, 1, .5, 0)$$

A

2D discrete F^0-transform

- Direct F-transform.
- Inverse F-transform.

$$F^0_{kl} = \frac{\sum_{x=0}^{A_w} \sum_{y=0}^{B_w} i_{kl}(x,y) g(x,y)}{\sum_{x=0}^{A_w} \sum_{y=0}^{B_w} g(x,y)}$$

2D discrete F^0-transform

- Direct F-transform.
- Inverse F-transform.

$$O^0(x,y) = \sum_{k=0}^{m} \sum_{l=0}^{n} F^0_{kl} A_k(x) B_l(y)$$



2D discrete F^1-transform

- Direct F-transform.
- Inverse F-transform.

$$F^1_{kl}(x,y) = c^{00}_{kl} + c^{10}_{kl}(x - x_k) + c^{01}_{kl}(x - y_l)$$

$$c^{00}_{kl} = \frac{\sum_{x=0}^{A_w}\sum_{y=0}^{B_w} i_{kl}(x,y)g(x,y)}{\sum_{x=0}^{A_w}\sum_{y=0}^{B_w} g(x,y)}$$

$$c^{10}_{kl} = \frac{\sum_{x=0}^{A_w}\sum_{y=0}^{B_w} i_{kl}(x,y)(x - x_k)g(x,y)}{\sum_{x=0}^{A_w}\sum_{y=0}^{B_w}(x - x_k)^2 g(x,y)}$$

$$c^{01}_{kl} = \frac{\sum_{x=0}^{A_w}\sum_{y=0}^{B_w} i_{kl}(x,y)(y - y_l)g(x,y)}{\sum_{x=0}^{A_w}\sum_{y=0}^{B_w}(y - y_l)^2 g(x,y)}$$

2D discrete F^1-transform

- Direct F-transform.
- Inverse F-transform.

2D discrete F^1-transform

- Direct F-transform.
- Inverse F-transform.

$$O^1(x,y) = \sum_{k=0}^{m} \sum_{l=0}^{n} F^1_{kl} A_k(x) B_l(y)$$

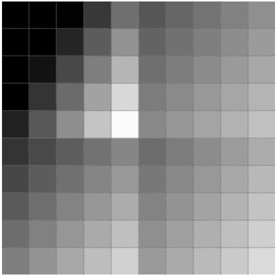


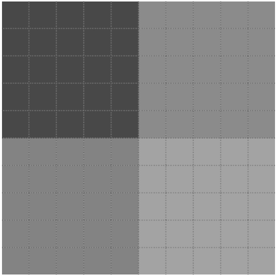
Comparison

- The output images.
- The components.

Comparison

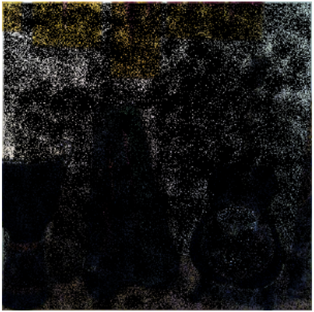
- The output images.
- The components.

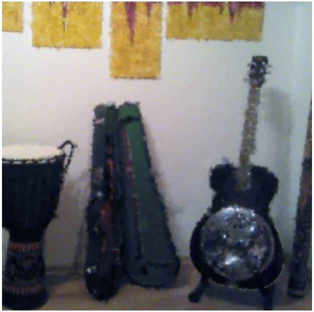




Applications

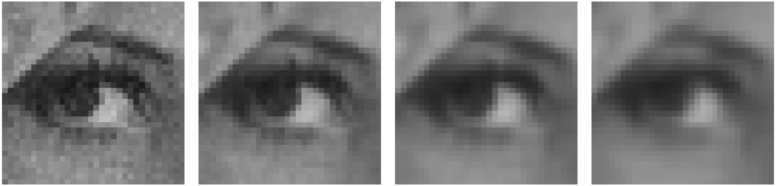
- Inpainting.
- Denoising.
- Upsampling.
- Filtering.
- Image creation.
- Edge detection.





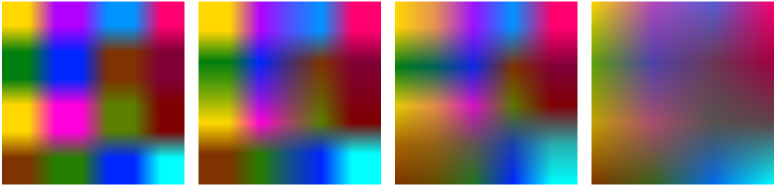
Applications

- Inpainting.
- Denoising.
- Upsampling.
- Filtering.
- Image creation.
- Edge detection.



Applications

- Inpainting.
- Denoising.
- Upsampling.
- Filtering.
- Image creation.
- Edge detection.



Applications

- Inpainting.
- Denoising.
- Upsampling.
- Filtering.
- Image creation.
- Edge detection.

Applications

- Inpainting.
- Denoising.
- Upsampling.
- Filtering.
- Image creation.
- Edge detection.

Applications

- Inpainting.
- Denoising.
- Upsampling.
- Filtering.
- Image creation.
- Edge detection.

Experiments

HOG descriptors

1. Setting the constants and directories containing training and testing data.
2. Loading the positive and negative training data and their respective labels.
3. Initializing the HOG descriptor with given parameters and calculating the descriptors for all training examples.
4. Initializing the SVM and train it using the given training data and respective labels.

F-transform descriptors

1. Setting the constants for the algorithm.
2. Loading the positive and negative training data and their respective labels.
3. Calculating the F-transform components for all training examples.
4. Initializing the SVM and train it using the given training data and respective labels.

Experiments

- Training data consist of 1000 images of size 60x160 px.
  - Positive examples - 500 containing a pedestrian
  - Negative examples - 500 without a pedestrian
- Testing data consist of 500 images of size 60x160 px.
  - Positive examples - 250 containing a pedestrian
  - Negative examples - 250 without a pedestrian
- Accuracy is computed as follows:
  1. load all positive and negative testing examples and calculate either their HOG descriptors and F-transform descriptors,
  2. iterate through positive and negative testing examples and use our trained SVM to predict the result,
  3. if it is correct, increment respective counter,
  4. aggregate the results and print out to console.

IRAFM

Experiments

- Training data consist of 1000 images of size 60x160 px.
  - Positive examples - 500 containing a pedestrian
  - Negative examples - 500 without a pedestrian
- Testing data consist of 500 images of size 60x160 px.
  - Positive examples - 250 containing a pedestrian
  - Negative examples - 250 without a pedestrian
- Accuracy is computed as follows:
  1. load all positive and negative testing examples and calculate either their HOG descriptors and F-transform descriptors,
  2. iterate through positive and negative testing examples and use our trained SVM to predict the result,
  3. if it is correct, increment respective counter,
  4. aggregate the results and print out to console.

IRAFM

Experiments

- Training data consist of 1000 images of size 60x160 px.
  - Positive examples - 500 containing a pedestrian
  - Negative examples - 500 without a pedestrian
- Testing data consist of 500 images of size 60x160 px.
  - Positive examples - 250 containing a pedestrian
  - Negative examples - 250 without a pedestrian
- Accuracy is computed as follows:
  1. load all positive and negative testing examples and calculate either their HOG descriptors and F-transform descriptors,
  2. iterate through positive and negative testing examples and use our trained SVM to predict the result,
  3. if it is correct, increment respective counter,
  4. aggregate the results and print out to console.

Results

Measured accuracy and learning time of HOG descriptors.

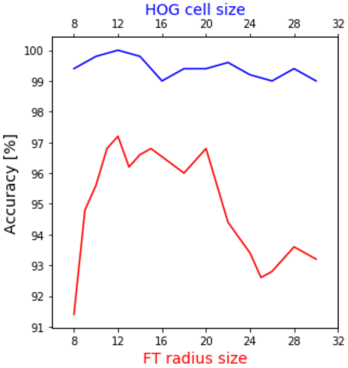
Cell size	SVM learning time [ms]	Accuracy [%]
8	227590.325	99.4
9	170642.722	99.6
10	162687.595	99.8
12	102559.346	100.0
14	60867.981	99.8
16	30479.193	99.0
18	22983.823	99.4
20	21601.116	99.4
22	9018.104	99.6
24	7708.107	99.2
26	7432.598	99.0
28	6425.163	99.4
30	6082.247	99.0

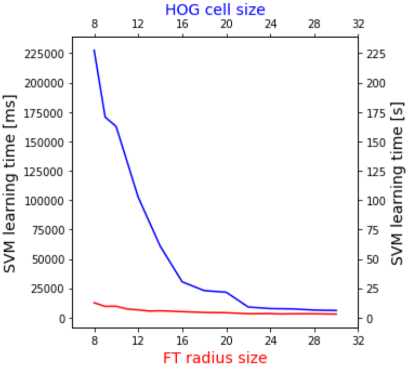
Results

Measured accuracy and learning time of F-transform descriptors.

Radius size	SVM learning time [ms]	Accuracy [%]
8	12643.460	91.4
9	9544.399	94.8
10	9701.500	95.6
11	7300.129	96.8
12	6641.180	97.2
13	5588.764	96.2
14	5799.976	96.6
15	5450.073	96.8
18	4383.862	96.0
20	4175.740	96.8
22	3276.331	94.4
24	3379.141	93.4
25	3126.122	92.6
26	3252.264	92.8
28	3265.752	93.6
30	2996.050	93.2

IRAFM


HOG cell size
8
12
16
20
24
28
32
100
99
98
97
96
95
94
93
92
91
Accuracy [%]
FT radius size


HOG cell size
8
12
16
20
24
28
32
225000
200000
175000
150000
125000
100000
75000
50000
25000
0
SVM learning time [ms]
225
200
175
150
125
100
75
50
25
0
SVM learning time [s]
FT radius size

Thank you for your attention!

Machine learning and fuzzy transform

Pavel Vlašánek

Institute for Research and Applications of Fuzzy Modeling
University of Ostrava

`pavel.vlasanek@osu.cz`

Seminář strojového učení a modelování

IRAFM