**Institut für Mess- und Regelungstechnik
mit Maschinenlaboratorium
Karlsruher Institut für Technologie
(KIT)
Prof. Dr.-Ing. C. Stiller**

Solutions for exam “Machine Vision” July 27, 2021

Question 1 (5+2 points)

Assume a point spread function

$$p(x) = \begin{cases} \frac{1}{5} & \text{if } 0 < x < 5 \\ 0 & \text{otherwise} \end{cases}$$

(a) Calculate the Fourier transform of p. Provide also intermediate steps of your calculation.

Remark: You may use the fact that the Fourier transform of the unit pulse

$$h(x) = \begin{cases} 1 & \text{if } -\frac{1}{2} < x < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

is the sinc function $sinc(k) = \frac{\sin(\pi k)}{\pi k}$

(b) Calculate the Wiener filter that can be used to eliminate blur created by the point spread function p. Assume that the image does not suffer from additional noise. Provide the Wiener filter in terms of its Fourier transform.

Solution

(a) Version 1:
Obviously, p is a scaled and shifted version of h, i.e. $p(x) = \frac{1}{5} \cdot h(\frac{x}{5} - \frac{1}{2})$. Hence,

$$\mathcal{F}\{p\}(k) = \frac{1}{5} \cdot e^{-2\pi i \frac{1}{2} 5k} \cdot 5\mathcal{F}\{h\}(5k) = e^{-5\pi ik} \frac{\sin(5\pi k)}{5\pi k}$$

Version 2:

$$\mathcal{F}\{p\}(k) = \int_{-\infty}^{\infty} p(x) e^{-2\pi ikx} dx = \frac{1}{5} \int_{0}^{5} e^{-2\pi ikx} dx = \frac{1}{-10\pi ik} \left[e^{-2\pi ikx} \right]_0^5$$

$$= \frac{1}{10\pi ik}(1 - e^{-10\pi ik}) = \frac{e^{-5\pi ik}}{5\pi k} \frac{e^{5\pi ik} - e^{-5\pi ik}}{2i} = e^{-5\pi ik} \frac{\sin(5\pi k)}{5\pi k}$$

(b) Since there is no noise we obtain the Fourier transform of the Wiener filter as reciprocal of the Fourier transform of the point spread function, i.e.

$$\mathcal{F}\{f\}(k) = \frac{1}{\mathcal{F}\{p\}(k)} = e^{5\pi ik} \frac{5\pi k}{\sin(5\pi k)}$$

Question 2 (3+3 points)

Assume that we want to perform edge detection on a rather dark image. In our first trial we apply the Canny operator with certain threshold values θ_L and θ_U. In a second trial we first apply contrast enhancement to improve the appearance of the image by applying the function

$$g \mapsto \max\{0, \min\{4 \cdot (g - 20), 100\}\}$$

to each pixel. We assume that gray values are encoded with numbers between 0 and 100.

(a) How do we need to adjust the parameters θ_L and θ_U of the Canny operator in the second trial to achieve approximately the same result of edge detection as in the first trial?

(b) In which gray value intervals do the results of the edge detection in the first and second trial differ? Justify your answer briefly.

Solution

(a) All gray values between 20 and 45 are stretched by a factor of 4 by the contrast enhancement. Hence, the gradient length also becomes four times as large. Therefore, the threshold values in the second trial must be four times as large as in the first trial.

(b) Gray values of less than 20 are all mapped to 0, gray values of more than 45 are all mapped to 100 by the contrast enhancement. Hence, all details vanish in these areas including all edges.

Question 3 (6 points)

The Matlab code below implements the total least sum of squares algorithm to fit a straight line to a set of points in the two-dimensional plane. The argument `x` is a vector of the x-coordinates of all points. The argument `y` is a vector of the y-coordinates of all points. The return value `n` should contain the unit normal vector of the line, the return value `c` contains the offset parameter of the line. Unfortunately, the code contains three logical errors. Find those errors and replace them by proper code.
You get one point for finding each error and one point for fixing it. Adding errors to the code or marking a proper part as erroneous part will lead to minus points. The whole task will be awarded with at least zero points.

Remark: The Matlab function `[V D] = eig(M)` calculates the Eigenvalues and unit Eigenvectors of a square matrix `M`. It returns the diagonal matrix `D` that contains the Eigenvalues in its diagonal, and the matrix of Eigenvectors `V`. The i-th column of `V` is the Eigenvector with respect to the Eigenvalue at the i-th diagonal position of `D`.

```
function [n,c] = tls(x, y)
  k = length(x)+length(y);
  sum_x = sum(x);
  sum_y = sum(y);
  sum_xx = sum(x.^2);
  sum_yy = sum(y.^2);
  sum_xy = sum(x.*y);
  [V D] = eig([sum_xx-sum_x*sum_x/k, sum_xy-sum_x*sum_y/k;
               sum_xy-sum_x*sum_y/k, sum_yy-sum_y*sum_y/k]);
  if (D(1,1)>D(2,2))
    n = V(:,1);
  else
    n = V(:,2);
  end
  c = -n(1)*sum_x-n(2)*sum_y/k;
end
```

Solution

```
function [n,c] = tls(x, y)
  k = length(x);
  sum_x = sum(x);
  sum_y = sum(y);
  sum_xx = sum(x.^2);
  sum_yy = sum(y.^2);
  sum_xy = sum(x.*y);
  [V D] = eig([sum_xx-sum_x*sum_x/k, sum_xy-sum_x*sum_y/k;
               sum_xy-sum_x*sum_y/k, sum_yy-sum_y*sum_y/k]);
  if (D(1,1)<D(2,2))
    n = V(:,1);
  else
    n = V(:,2);
  end
  c = -n(1)*sum_x/k-n(2)*sum_y/k;
end
```

Question 4 (5 points)

Does *k-means clustering* always converge to the same solution if the prototype colors are initialized randomly? If *yes*, provide a formal proof, if *no* provide a counter example.

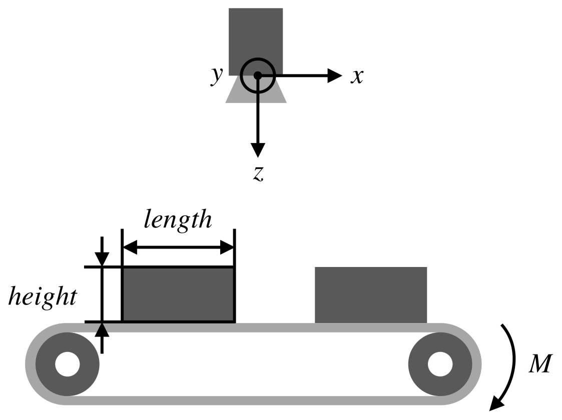
Solution

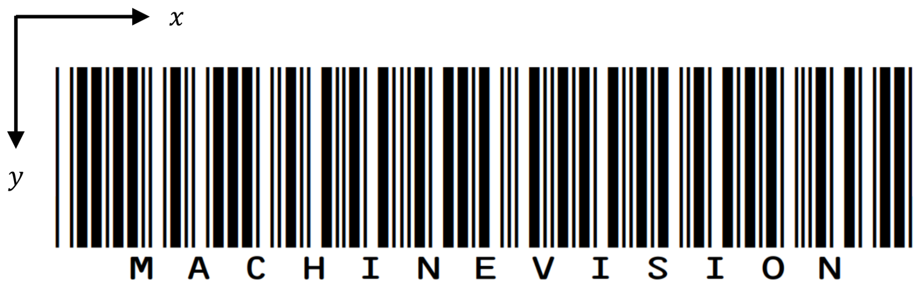
No, it does not always converge to the same result. We provide a simple counter example with three graylevel pixels $g_1 = 0$, $g_2 = 1$, and $g_3 = 2$ and $k = 2$. If we initialize the prototype colors c_1, c_2 with $c_1 = 0.5$ and $c_2 = 2$ k-means converges immediately and we obtain the partitioning into the two clusters $\{g_1, g_2\}$ and $\{g_3\}$. If we initialize the prototype colors with $c_1 = 0$ and $c_2 = 1.5$ k-means also converges immediately and we obtain the two clusters $\{g_1\}$, $\{g_2, g_3\}$.

Question 5 (4 + 3 + 2 points)

A camera is mounted straight above a conveyor belt. The intrinsic parameters of the camera are given by the matrix **A**. An image processing algorithm is checking the barcodes on top of the boxes, which are moving on the conveyor belt. All barcodes are aligned in x-direction as shown in the figure below. The boxes with the barcode attached move in horizontal direction. The vertical distance between the barcodes and the origin of the camera coordinate system is 100 mm. The thinest lines in the barcode have a width of 0.5 mm. For simplification, a pinhole camera model is assumed.

$$\mathbf{A} = \begin{pmatrix} 1000 & 0 & 1500 \\ 0 & 1000 & 800 \\ 0 & 0 & 1 \end{pmatrix}$$





(a) Determine the edge length of the square on top of the boxes, that is covered by a single pixel in the camera image. Provide your results in millimeters.

(b) Since the conveyor belt is moving with $2\frac{\text{m}}{\text{s}}$ motion blur affects the image. Calculate the maximum exposure time to obtain a sufficiently sharp image, i.e. an image in which two neighboring bars are not intersecting.

(c) How can the setup in (a) be changed to neglect motion blur and allow higher exposure times? Justify your answer briefly.

Remark: Part (c) can be solved independently of part (a) and (b)

Solution

(a)

$$\Delta x = \frac{z \cdot \Delta u}{\alpha'} = \frac{100 \text{ mm} \cdot 1}{1000} = 0.1 \text{ mm}$$

(b)

$$t = \frac{s}{v} = \frac{0.4 \text{ mm}}{2 \frac{\text{m}}{\text{s}}} = 200\ \mu\text{s}$$

(c) Turn boxes or barcodes by $\pm 90°$.

Question 6 (3+2 points)

Assume that we trained an ensemble classifier with AdaBoost. The ensemble consists of six binary classifiers $c_1, \ldots, c_6$. The ensemble is applied to three new patterns $\vec{x}_1, \vec{x}_2, \vec{x}_3$. The table below shows the classification result of each of the six classifiers for each of the three patterns. The column β_i shows the classifier weight of each classifier.

	β_i	$c_i(\vec{x}_1)$	$c_i(\vec{x}_2)$	$c_i(\vec{x}_3)$	$c_i(\vec{x}_4)$
c_1	$\frac{6}{20}$	$+$	$-$	$+$	
c_2	$\frac{5}{20}$	$-$	$-$	$+$	
c_3	$\frac{3}{20}$	$-$	$-$	$-$	
c_4	$\frac{3}{20}$	$+$	$+$	$-$	
c_5	$\frac{1}{20}$	$+$	$+$	$-$	
c_6	$\frac{2}{20}$	$+$	$+$	$-$	
ensemble					*undecided*

(a) Provide the classification result of the ensemble for each of the three patterns. You may provide your answers in the last row of the table.

(b) Create an example in which the ensemble classifier is undecided. For that example provide the individual decisions of each of the six classifiers $c_1, \ldots, c_6$. You may provide your answer in the last column of the table named $c_i(\vec{x}_4)$.

Solution

	β_i	$c_i(\vec{x}_1)$	$c_i(\vec{x}_2)$	$c_i(\vec{x}_3)$	$c_i(\vec{x}_4)$
c_1	$\frac{6}{20}$	$+$	$-$	$+$	$+$
c_2	$\frac{5}{20}$	$-$	$-$	$+$	$-$
c_3	$\frac{3}{20}$	$-$	$-$	$-$	$+$
c_4	$\frac{3}{20}$	$+$	$+$	$-$	$-$
c_5	$\frac{1}{20}$	$+$	$+$	$-$	$+$
c_6	$\frac{2}{20}$	$+$	$+$	$-$	$-$
ensemble		$+$	$-$	$+$	*undecided*

Question 7 (6+6 points)

In this task a linear classifier should be developed that can be used to recognize gray level corners in binarized images, in which possible gray values are only 0 (black) or 1 (white). The classifier is applied to each 2-by-2 block of pixels and should return either 1 if the 2-by-2 block shows a corner, or -1 if the 2-by-2 block does not show a corner. The classifier should use patterns which extract certain features from the 2-by-2 block. Below you find four 2-by-2 blocks $x^{(i)}$ with the appropriate expected classification result $d^{(i)}$.

$$x^{(1)} : \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 1 \\ \hline \end{array} \qquad x^{(2)} : \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} \qquad x^{(3)} : \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 1 \\ \hline \end{array} \qquad x^{(4)} : \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 0 & 1 \\ \hline \end{array}$$

$$d^{(1)} = -1 \qquad d^{(2)} = -1 \qquad d^{(3)} = +1 \qquad d^{(4)} = +1$$

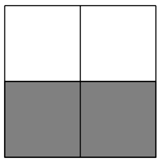
We consider three possible features for this task as follows.

(i) The first approach uses only the average gray value as single feature.

(ii) The second approach segments the 2-by-2 block into segments using the *connected components labeling* algorithm. It creates two features, the number of segments, and the number of pixels of the largest segment.

(iii) The third approach applies a Haar feature to the 2-by-2 block, see the drawing of the Haar feature below. The Haar feature is calculated by determining the average gray value in the bright area minus the average gray value in the dark area.

Haar feature :



Solve the following tasks

(a) Determine the training patterns for each of the four example blocks and each of the three classification approaches.

(b) For each classification approach check whether a linear classifier is able to classify the four training examples error-free. For each classifier either provide such a linear classifier or provide an argument why such a classifier does not exist.

Solution

Average grey value:

$x_{c1}^{(1)} = 0.5$

$x_{c1}^{(2)} = 0$

$x_{c1}^{(3)} = 0.25$

$x_{c1}^{(4)} = 0.75$

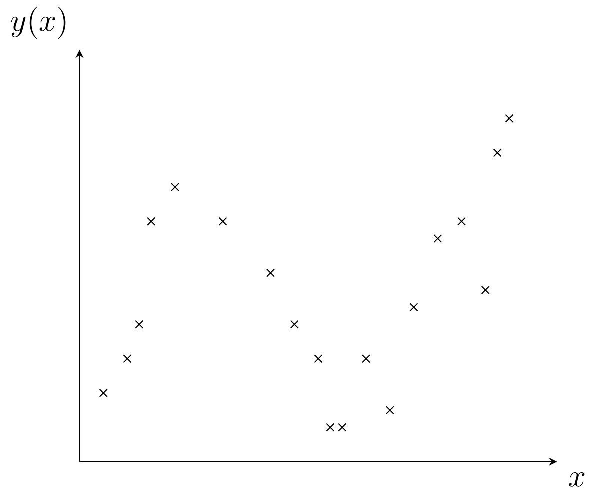
A linear classifier cannot be used since 0.5 is between 0.25 and 0.75

Connected component labeling (size of the largest segment in pixel, number of segments):
$x_{c2}^{(1)} = (2,2)^\top$
$x_{c2}^{(2)} = (1,4)^\top$
$x_{c2}^{(3)} = (2,3)^\top$
$x_{c2}^{(4)} = (2,3)^\top$
A linear classifier can be used, e.g. $x(2) > -\frac{1}{3} * x(1) + \frac{8}{3}$

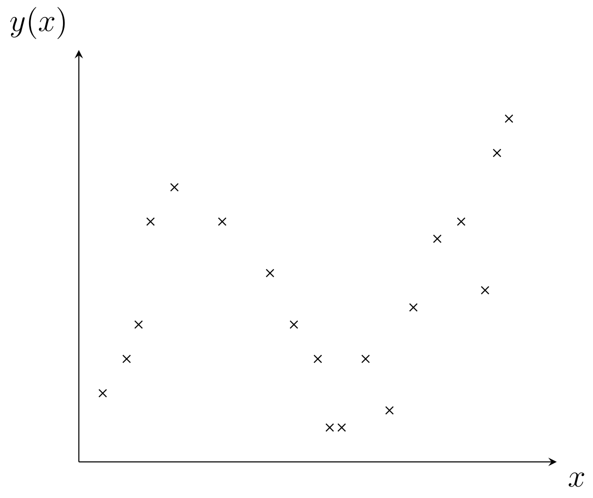
Haar features:
$x_{c3}^{(1)} = -1$
$x_{c3}^{(2)} = 0$
$x_{c3}^{(3)} = -\frac{1}{2}$
$x_{c3}^{(4)} = \frac{1}{2}$
A linear classifier cannot be used since 0 is between $-\frac{1}{2}$ and $+\frac{1}{2}$

Question 8 (6 points)

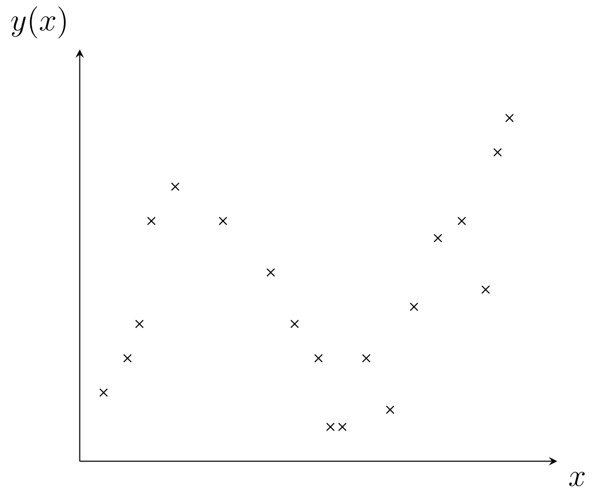
A neural network should be used to fit a curve through a set of training patterns with one-dimensional input and one-dimensional, real-valued output. In the graphs below the training set is shown, the training patterns are visualized as crosses. Draw a possible resulting curve for an underfitting network (a), an overfitting network (b) and well generalizing network (c) into the graphs below. Draw carefully!



(a) curve fitting by an **underfitting** neural network.

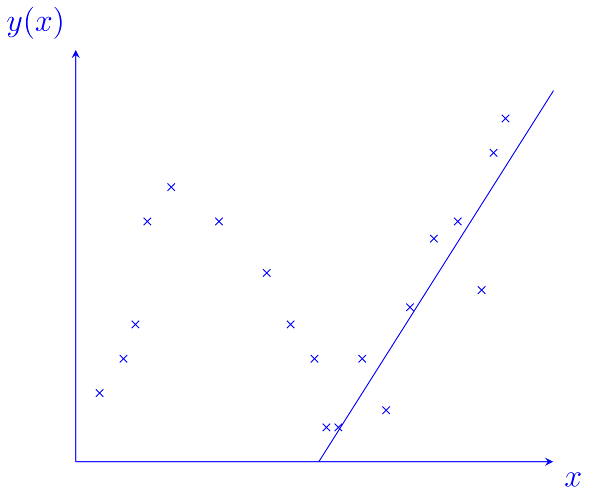


(b) curve fitting by an **overfitting** neural network.

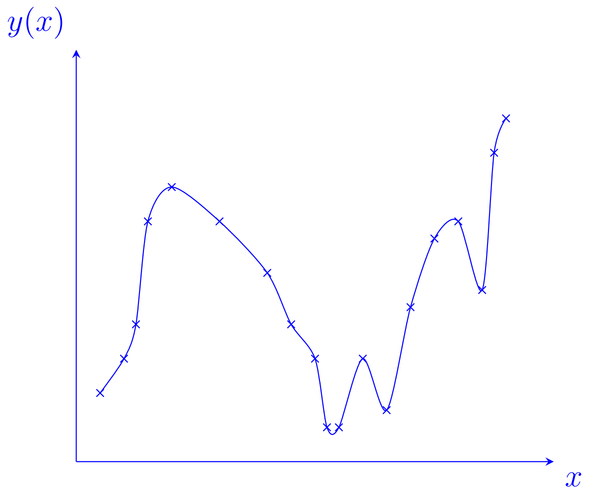


(c) curve fitting by a well **generalizing** neural network.

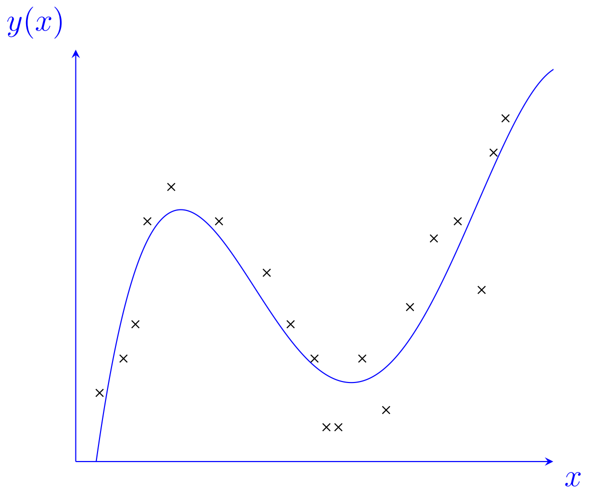
Solution



(a) curve fitting by an **underfitting** neural network.



(b) curve fitting by an **overfitting** neural network.



(c) curve fitting by a **generalizing** neural network.

Question 9 (4 points)

Consider a *Gated Recurrent Unit (GRU)* as part of a recurrent neural network. Assume that the dimension of the state vector $\vec{s}$ is 1 and the dimension of the input vector $\vec{x}$ is 9. How many weights must be trained for a single GRU?

Solution

- reset gate: 9 weights for input $\vec{x}$, 1 weight for $\vec{s}$, 1 bias weight, in total 11 weights
- update gate: 9 weights for input $\vec{x}$, 1 weight for $\vec{s}$, 1 bias weight, in total 11 weights
- calculation of update value: 9 weights for input $\vec{x}$, 1 weight for $\vec{s}$, 1 bias weight, in total 11 weights

In total: $11 + 11 + 11 = 33$ weights

Gesamtpunkte: 60