Data Analytics and Machine Learning Group
Department of Informatics
Technical University of Munich

Eexam
Place student sticker here

Note:

- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Machine Learning

Graded Exercise:	IN2064 / Retake	**Date:**	Thursday 1st April, 2021
Examiner:	Prof. Dr. Stephan Günnemann	**Time:**	16:30 – 18:30

Working instructions

- This graded exercise consists of **34 pages** with a total of **11 problems**.
  Please make sure now that you received a complete copy of the graded exercise.
- The total amount of achievable credits in this graded exercise is 40.
- This document is copyrighted and it is **illegal** for you to distribute it or upload it to any third-party websites.
- Do **not** submit the problem descriptions (this document) to TUMexam

Left room from _____ to _____ / Early submission at _____

Problem 1: Probabilistic Inference (Version A) (4 credits)

We have observed N coin flips of which T landed tails and H landed heads ($T + H = N$). We model each coin flip with a Bernoulli(θ) distribution with a shared unknown probability θ of coming up heads, i.e. p(heads $\mid \theta$) = θ.

Furthermore, we assume that θ follows a Beta(a, b) distribution with parameters $a > 0$ and $b > 0$. In this problem we are not interested in estimating θ, but rather estimating the parameters a and b.

0 1 2

a) Is it possible to compute a maximum likelihood estimate (MLE) $\arg\max_{a,b} p(\mathcal{D} \mid a, b)$ of a and b in this model?
If yes, briefly describe a way to do it and how the variable θ is handled.
If no, explain why it is not possible and the role of the variable θ.

0 1 2

b) Is it possible to compute a maximum a posteriori (MAP) estimate $\arg\max_{a,b} p(a, b \mid \mathcal{D})$ of a and b in this model?
If yes, briefly describe how to compute the MAP estimate of a and b.
If no, describe how and why the model would need to be changed/extended to allow MAP estimation of a and b.

a) Yes. But unobserved parameter θ needs to be marginalized out first

$$p(D \mid a,b) = \int p(D \mid \theta) \cdot p(\theta \mid a,b)\, d\theta = \mathbb{E}_{p(\theta|a,b)}[p(D \mid \theta)]$$

b)
No, it is not immediately possible because for MAP estimation you need a prior on the variables you want to infer. So you would need to introduce a hyperprior on the parameters a and b. Then you can compute an MAP by maximizing $p(D \mid a,b)\, p(a,b)$ where the data likelihood is computed in a).

Problem 2: Decision Trees (Version A) (5 credits)

0 1

a) Suppose we randomly sample subsets of features to learn separate trees which are then combined. What is this technique called?

Bagging at feature level

0 1 2

b) We have a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ of N instances with $\boldsymbol{x}_i \in \mathbb{R}^2$ and $y_i \in \{0, 1\}$. We aim to train a decision tree using entropy as the splitting criterion. We stop building the tree when there is zero *improvement* in purity for all splits.

Specify a small dataset $\mathcal{D}$ so that the learned decision tree has no splits – the root node is a leaf. Write down all $(\boldsymbol{x}_i, y_i)$ tuples in your $\mathcal{D}$ and make sure it contains at least one instance from each class. Justify your answer.

Hint: you do not need more than a few instances.

XOR dataset

$D = \{([a,b], 0), ([a,b], 1)\}$

2 instances with exactly the same features but different labels

0 1 2

c) Draw the decision tree corresponding to the decision boundaries shown on Figure 3.1 where $0 \leq a, b, c, d \leq 1$ are some arbitrary constants and there are four classes marked with four numbers and different colors. Assume $x_1, x_2 \in [0, 1]$.

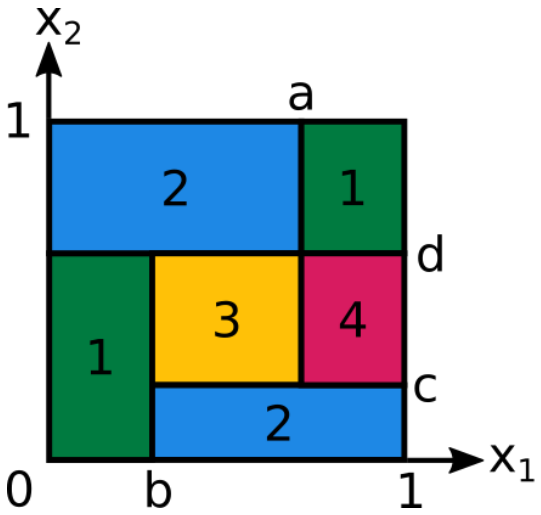
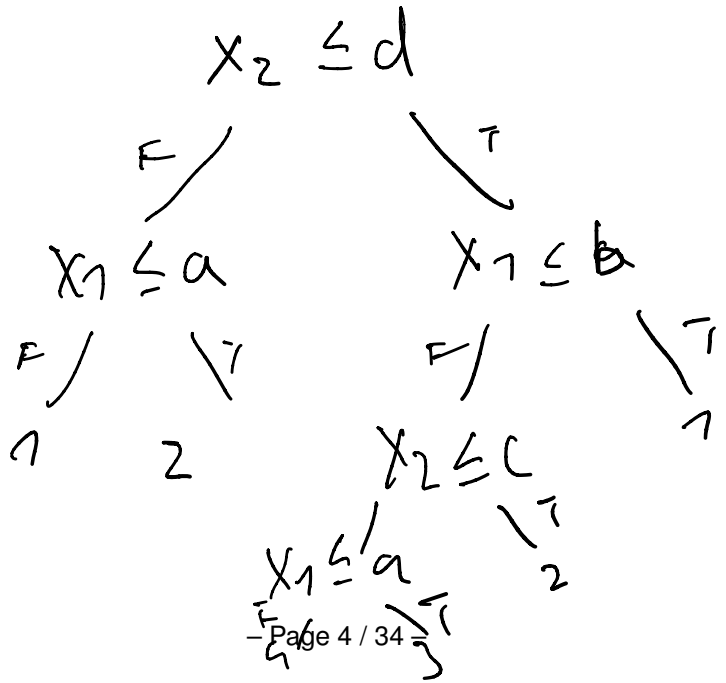


Figure 3.1: Decision boundaries.



Problem 3: Linear Regression (Version A) (2 credits)

0 1 2

We have a dataset $\{(\boldsymbol{x}_i, y_i) \mid \boldsymbol{x}_i \in \mathbb{R}^2, y_i \in \mathbb{R}\}_{i=1}^N$ and want to fit the following model with three parameters $\boldsymbol{w} = \begin{pmatrix} a & b & c \end{pmatrix}^\mathsf{T} \in \mathbb{R}^3$ to it.

$$f(\boldsymbol{x}, \boldsymbol{w}) = a \sin(\boldsymbol{x}_2) + \frac{1}{2} b \|\boldsymbol{x}\|_1 - \boldsymbol{x}_1^2 \boldsymbol{x}_2 c$$

Give a closed form expression for the optimal $\boldsymbol{w}$ minimizing the squared error between the predictions and targets

$$\sum_{i=1}^{N} (f(\boldsymbol{x}_i, \boldsymbol{w}) - y_i)^2 .$$

Justify your answer.

Note: You can use results from the lecture without deriving them again.

f is linear in the parameters $w = (a\ b\ c)^T$

$$f(x, w) = w^T \begin{pmatrix} \sin(x_2) \\ 1/2 \|x\|_1 \\ -x_1^2 x_2 \end{pmatrix}$$

We define a feature transformation

$$\phi(x) = \begin{pmatrix} \sin(x_2) \\ 1/2 \|x\|_1 \\ -x_1^2 x_2 \end{pmatrix}$$

$$y = \Phi w$$

Apply the closed form for ordinary least squares and get

$$w^* = (\Phi^T \Phi)^{-1} \Phi^T y$$

Problem 4: Logistic Regression (Version A) (6 credits)

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \mid \boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \{0, 1\}\}$ be a classification dataset that is *not* linearly separable. Furthermore, let $\mathcal{D}_\alpha = \{(\alpha \boldsymbol{x}_i, y_i) \mid (\boldsymbol{x}_i, y_i) \in \mathcal{D}\}$ be a scaled copy of $\mathcal{D}$ with $\alpha > 1$. $f(\boldsymbol{x}, \boldsymbol{w}) : \mathbb{R}^m \times \mathbb{R}^m \to [0, 1]$ is a logistic regression model on $\mathbb{R}^m$ for some m with parameters $\boldsymbol{w}$. $f(\boldsymbol{x}, \boldsymbol{w})$ outputs the predicted probabilities for class 1.

You train two logistic regression models on $\mathcal{D}$ and $\mathcal{D}_\alpha$ *without regularization* and obtain the optimal parameters $\boldsymbol{w}^*$ and $\boldsymbol{w}^{*,\alpha}$, respectively. Consider a test point $\boldsymbol{x}_{\text{test}} \in \mathbb{R}^d$ and the predicted probabilities by the two models, $s = f(\boldsymbol{x}_{\text{test}}, \boldsymbol{w}^*)$ and $t = f(\boldsymbol{x}_{\text{test}}, \boldsymbol{w}^{*,\alpha})$.

a) Is $s > t$ possible? Is $s = t$ possible? Is $s < t$ possible? Justify your answers.

0
1
2

Now consider another dataset $\mathcal{D}_3$ and a logistic regression model $f(\boldsymbol{x}, \boldsymbol{w})$ on $\mathcal{D}_3$.

$$\mathcal{D}_3 = \{(\boldsymbol{x}_i, 1) \mid \boldsymbol{x}_i \in \mathbb{R}^2, \boldsymbol{x}_{i,1} > 0, \boldsymbol{x}_{i,2} > 0\} \cup \{(\boldsymbol{x}_i, 0) \mid \boldsymbol{x}_i \in \mathbb{R}^2, \boldsymbol{x}_{i,1} < 0, \boldsymbol{x}_{i,2} < 0\}$$

In the following, treat ∞ as an actual value, i.e. $\infty \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ would be a vector in the same direction as $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ but with infinite length/norm.

b) Briefly explain why the maximum likelihood estimate of $\boldsymbol{w}$ on $\mathcal{D}_3$ obtained by training *without* regularization is not unique in this setting.

0
1
2

c) Give two possible maximum likelihood estimates (without regularization) of the parameters $\boldsymbol{w}$ on $\mathcal{D}_3$, $\boldsymbol{w}^{*,a}$ and $\boldsymbol{w}^{*,b}$, and a test point $\boldsymbol{x}_{\text{test}}$ such that either $f(\boldsymbol{x}_{\text{test}}, \boldsymbol{w}^{*,a}) < \frac{1}{2} < f(\boldsymbol{x}_{\text{test}}, \boldsymbol{w}^{*,b})$ or $f(\boldsymbol{x}_{\text{test}}, \boldsymbol{w}^{*,a}) > \frac{1}{2} > f(\boldsymbol{x}_{\text{test}}, \boldsymbol{w}^{*,b})$ holds. Justify your answer.

0
1
2

Problem 4: Logistic Regression (Version B) (6 credits)

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \mid \boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \{0, 1\}\}$ be a classification dataset that is *not* linearly separable. Furthermore, let $\mathcal{D}_\alpha = \{(\alpha \boldsymbol{x}_i, y_i) \mid (\boldsymbol{x}_i, y_i) \in \mathcal{D}\}$ be a scaled copy of $\mathcal{D}$ with $\alpha > 1$. $f(\boldsymbol{x}, \boldsymbol{w}) : \mathbb{R}^m \times \mathbb{R}^m \to [0, 1]$ is a logistic regression model on $\mathbb{R}^m$ for some m with parameters $\boldsymbol{w}$. $f(\boldsymbol{x}, \boldsymbol{w})$ outputs the predicted probabilities for class 1.

You train two logistic regression models on $\mathcal{D}$ and $\mathcal{D}_\alpha$ *without regularization* and obtain the optimal parameters $\boldsymbol{w}^*$ and $\boldsymbol{w}^{*,\alpha}$, respectively. Consider a test point $\boldsymbol{x}_{\text{test}} \in \mathbb{R}^d$ and the predicted probabilities by the two models, $s = f(\boldsymbol{x}_{\text{test}}, \boldsymbol{w}^*)$ and $t = f(\boldsymbol{x}_{\text{test}}, \boldsymbol{w}^{*,\alpha})$.

0 1 2

a) Is $s > t$ possible? Is $s = t$ possible? Is $s < t$ possible? Justify your answers.

Now consider another dataset $\mathcal{D}_3$ and a logistic regression model $f(\boldsymbol{x}, \boldsymbol{w})$ on $\mathcal{D}_3$.

$$\mathcal{D}_3 = \{(\boldsymbol{x}_i, 1) \mid \boldsymbol{x}_i \in \mathbb{R}^2, \boldsymbol{x}_{i,1} > 0, \boldsymbol{x}_{i,2} > 0\} \cup \{(\boldsymbol{x}_i, 0) \mid \boldsymbol{x}_i \in \mathbb{R}^2, \boldsymbol{x}_{i,1} < 0, \boldsymbol{x}_{i,2} < 0\}$$

In the following, treat ∞ as an actual value, i.e. $\infty \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ would be a vector in the same direction as $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ but with infinite length/norm.

0 1 2

b) Briefly explain why the maximum likelihood estimate of $\boldsymbol{w}$ on $\mathcal{D}_3$ obtained by training *without* regularization is not unique in this setting.

0 1 2

c) Give two possible maximum likelihood estimates (without regularization) of the parameters $\boldsymbol{w}$ on $\mathcal{D}_3$, $\boldsymbol{w}^{*,a}$ and $\boldsymbol{w}^{*,b}$, and a test point $\boldsymbol{x}_{\text{test}}$ such that either $f(\boldsymbol{x}_{\text{test}}, \boldsymbol{w}^{*,a}) < \frac{1}{2} < f(\boldsymbol{x}_{\text{test}}, \boldsymbol{w}^{*,b})$ or $f(\boldsymbol{x}_{\text{test}}, \boldsymbol{w}^{*,a}) > \frac{1}{2} > f(\boldsymbol{x}_{\text{test}}, \boldsymbol{w}^{*,b})$ holds. Justify your answer.

Problem 5: Optimization (Version A) (3 credits)

Suppose we're minimizing some differentiable convex function $f : \mathbb{R}^d \to \mathbb{R}$ using one of the following variants of gradient descent. Let θ_t be the value of the parameter at iteration t, and θ_{t+1} be the value of the parameter at the iteration $t + 1$ for some integer t.

a) Suppose we are using gradient descent with line search.

Is the inequality $f(\theta_{t+1}) \leq f(\theta_t)$ guaranteed to always hold? Justify your answer.

b) Suppose we are using gradient descent with fixed step size.

Is the inequality $f(\theta_{t+1}) \leq f(\theta_t)$ guaranteed to always hold? Justify your answer.

c) Suppose we are using gradient descent with adaptive learning rate (Adam - Adaptive moment estimation).

Is the inequality $f(\theta_{t+1}) \leq f(\theta_t)$ guaranteed to always hold? Justify your answer.

Problem 5: Optimization (Version B) (3 credits)

Suppose we're minimizing some differentiable convex function $f : \mathbb{R}^d \to \mathbb{R}$ using one of the following variants of gradient descent. Let θ_t be the value of the parameter at iteration t, and θ_{t+1} be the value of the parameter at the iteration $t+1$ for some integer t.

0 ☐
1 ☐

a) Suppose we are using gradient descent with fixed step size.

Is the inequality $f(\theta_{t+1}) \leq f(\theta_t)$ guaranteed to always hold? Justify your answer.

0 ☐
1 ☐

b) Suppose we are using gradient descent with adaptive learning rate (Adam - Adaptive moment estimation).

Is the inequality $f(\theta_{t+1}) \leq f(\theta_t)$ guaranteed to always hold? Justify your answer.

0 ☐
1 ☐

c) Suppose we are using gradient descent with line search.

Is the inequality $f(\theta_{t+1}) \leq f(\theta_t)$ guaranteed to always hold? Justify your answer.

Problem 5: Optimization (Version C) (3 credits)

Suppose we're minimizing some differentiable convex function $f : \mathbb{R}^d \to \mathbb{R}$ using one of the following variants of gradient descent. Let θ_t be the value of the parameter at iteration t, and θ_{t+1} be the value of the parameter at the iteration $t + 1$ for some integer t.

a) Suppose we are using gradient descent with line search.

Is the inequality $f(\theta_{t+1}) \leq f(\theta_t)$ guaranteed to always hold? Justify your answer.

0
1

b) Suppose we are using gradient descent with adaptive learning rate (Adam - Adaptive moment estimation).

Is the inequality $f(\theta_{t+1}) \leq f(\theta_t)$ guaranteed to always hold? Justify your answer.

0
1

c) Suppose we are using gradient descent with fixed step size.

Is the inequality $f(\theta_{t+1}) \leq f(\theta_t)$ guaranteed to always hold? Justify your answer.

0
1

Problem 5: Optimization (Version D) (3 credits)

Suppose we're minimizing some differentiable convex function $f : \mathbb{R}^d \to \mathbb{R}$ using one of the following variants of gradient descent. Let θ_t be the value of the parameter at iteration t, and θ_{t+1} be the value of the parameter at the iteration $t+1$ for some integer t.

0 ☐
1 ☐

a) Suppose we are using gradient descent with adaptive learning rate (Adam - Adaptive moment estimation).

Is the inequality $f(\theta_{t+1}) \leq f(\theta_t)$ guaranteed to always hold? Justify your answer.

0 ☐
1 ☐

b) Suppose we are using gradient descent with fixed step size.

Is the inequality $f(\theta_{t+1}) \leq f(\theta_t)$ guaranteed to always hold? Justify your answer.

0 ☐
1 ☐

c) Suppose we are using gradient descent with line search.

Is the inequality $f(\theta_{t+1}) \leq f(\theta_t)$ guaranteed to always hold? Justify your answer.

Problem 6: Deep Learning (Version A) (3 credits)

Suppose $\boldsymbol{x} \in \mathbb{R}^N$ and $\boldsymbol{y} \in \mathbb{R}^N$ are two vectors. We define the function $f\colon \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ as

$$f(\boldsymbol{x}, \boldsymbol{y}) = \log\left(1 + \exp(\boldsymbol{x}^T \boldsymbol{y})\right).$$

The code below implements the computation of $f(\boldsymbol{x}, \boldsymbol{y})$ as well as its gradients w.r.t. $\boldsymbol{x}$ and $\boldsymbol{y}$ using backpropagation (similarly to how we did in Exercise sheet 7). However, some code fragments are missing. Your task is to complete the missing code fragments.

```
import numpy as np


class F:
    def forward(self, x, y):
        self.cache = (x, y)
        ####################################################
        # MISSING CODE FRAGMENT #1
        ####################################################
        return out

    def backward(self, d_out):
        # x, y are np.arrays of shape (N,)
        x, y = self.cache
        ####################################################
        # MISSING CODE FRAGMENT #2
        ####################################################
        return d_x, d_y


# Example usage
f = F()
x = np.array([1., 2., 3])
y = np.array([-2., 3., -1.])

z = f.forward(x, y)
d_z = 1.0
d_x, d_y = f.backward(d_z)
```

a) Complete the `MISSING CODE FRAGMENT #1`

0 1

```
out = np.log(1 + np.ex(x @ y))
```

b) Complete the `MISSING CODE FRAGMENT #2`

0 1 2

```
s = np.exp(x @ y) / (1 + np.exp(x @ y))

d_x = d_out * s * y

d_y = d_out * s * x
```

Problem 6: Deep Learning (Version B) (3 credits)

Suppose $\boldsymbol{x} \in \mathbb{R}^N$ and $\boldsymbol{y} \in \mathbb{R}^N$ are two vectors. We define the function $f\colon \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ as

$$f(\boldsymbol{x}, \boldsymbol{y}) = \log\left(\exp(\boldsymbol{x}^T \boldsymbol{y}) - 1\right).$$

The code below implements the computation of $f(\boldsymbol{x}, \boldsymbol{y})$ as well as its gradients w.r.t. $\boldsymbol{x}$ and $\boldsymbol{y}$ using backpropagation (similarly to how we did in Exercise sheet 7). However, some code fragments are missing. Your task is to complete the missing code fragments.

```
import numpy as np


class F:
    def forward(self, x, y):
        self.cache = (x, y)
        ####################################################
        # MISSING CODE FRAGMENT #1
        ####################################################
        return out

    def backward(self, d_out):
        # x, y are np.arrays of shape (N,)
        x, y = self.cache
        ####################################################
        # MISSING CODE FRAGMENT #2
        ####################################################
        return d_x, d_y


# Example usage
f = F()
x = np.array([1., 2., 3])
y = np.array([-2., 3., -1.])

z = f.forward(x, y)
d_z = 1.0
d_x, d_y = f.backward(d_z)
```

0 ☐
1 ☐

a) Complete the `MISSING CODE FRAGMENT #1`

0 ☐
1 ☐
2 ☐

b) Complete the `MISSING CODE FRAGMENT #2`

Problem 6: Deep Learning (Version C) (3 credits)

Suppose $\boldsymbol{x} \in \mathbb{R}^N$ and $\boldsymbol{y} \in \mathbb{R}^N$ are two vectors. We define the function $f\colon \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ as

$$f(\boldsymbol{x}, \boldsymbol{y}) = \log\left(1 + \exp(\boldsymbol{x}^T \boldsymbol{y})\right).$$

The code below implements the computation of $f(\boldsymbol{x}, \boldsymbol{y})$ as well as its gradients w.r.t. $\boldsymbol{x}$ and $\boldsymbol{y}$ using backpropagation (similarly to how we did in Exercise sheet 7). However, some code fragments are missing. Your task is to complete the missing code fragments.

```
import numpy as np


class F:
    def forward(self, x, y):
        self.cache = (x, y)
        ####################################################
        # MISSING CODE FRAGMENT #1
        ####################################################
        return out

    def backward(self, d_out):
        # x, y are np.arrays of shape (N,)
        x, y = self.cache
        ####################################################
        # MISSING CODE FRAGMENT #2
        ####################################################
        return d_x, d_y


# Example usage
f = F()
x = np.array([1., 2., 3])
y = np.array([-2., 3., -1.])

z = f.forward(x, y)
d_z = 1.0
d_x, d_y = f.backward(d_z)
```

a) Complete the `MISSING CODE FRAGMENT #1`

0
1

b) Complete the `MISSING CODE FRAGMENT #2`

0
1
2

Problem 6: Deep Learning (Version D) (3 credits)

Suppose $\boldsymbol{x} \in \mathbb{R}^N$ and $\boldsymbol{y} \in \mathbb{R}^N$ are two vectors. We define the function $f\colon \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ as

$$f(\boldsymbol{x}, \boldsymbol{y}) = \log\left(\exp(\boldsymbol{x}^T \boldsymbol{y}) - 1\right).$$

The code below implements the computation of $f(\boldsymbol{x}, \boldsymbol{y})$ as well as its gradients w.r.t. $\boldsymbol{x}$ and $\boldsymbol{y}$ using backpropagation (similarly to how we did in Exercise sheet 7). However, some code fragments are missing. Your task is to complete the missing code fragments.

```
import numpy as np


class F:
    def forward(self, x, y):
        self.cache = (x, y)
        ##################################################
        # MISSING CODE FRAGMENT #1
        ##################################################
        return out

    def backward(self, d_out):
        # x, y are np.arrays of shape (N,)
        x, y = self.cache
        ##################################################
        # MISSING CODE FRAGMENT #2
        ##################################################
        return d_x, d_y


# Example usage
f = F()
x = np.array([1., 2., 3])
y = np.array([-2., 3., -1.])

z = f.forward(x, y)
d_z = 1.0
d_x, d_y = f.backward(d_z)
```

0 1

a) Complete the `MISSING CODE FRAGMENT #1`

0 1 2

b) Complete the `MISSING CODE FRAGMENT #2`

Problem 7: Kernels (Version A) (3 credits)

Let $\mathbf{\Sigma} \in \mathbb{R}^{D \times D}$ be a given invertible, positive semi-definite matrix and $c \in \mathbb{R}$ be a given constant. Consider the kernel:

$$k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}, \quad k(\boldsymbol{x}_1, \boldsymbol{x}_2) = c^2 \exp\left(-\frac{1}{2}(\boldsymbol{x}_1 - \boldsymbol{x}_2)\mathbf{\Sigma}^{-1}(\boldsymbol{x}_1 - \boldsymbol{x}_2)\right) .$$

Prove or disprove that k is a valid kernel.

Hint: If $k_1(\boldsymbol{x}_1, \boldsymbol{x}_2)$ then $\exp(k_1(\boldsymbol{x}_1, \boldsymbol{x}_2))$ is also a kernel.

0
1
2
3

Problem 7: Kernels (Version B) (3 credits)

Let $\mathbf{\Sigma} \in \mathbb{R}^{D \times D}$ be a given invertible, positive semi-definite matrix and $a \in \mathbb{R}$ be a given constant. Consider the kernel:

$$k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}, \quad k(\boldsymbol{x}_1, \boldsymbol{x}_2) = a^2 \exp\left(-\frac{1}{2}(\boldsymbol{x}_1 - \boldsymbol{x}_2)\mathbf{\Sigma}^{-1}(\boldsymbol{x}_1 - \boldsymbol{x}_2)\right) .$$

0 ☐
1 ☐
2 ☐
3 ☐

Prove or disprove that k is a valid kernel.

Hint: If $k_1(\boldsymbol{x}_1, \boldsymbol{x}_2)$ then $\exp(k_1(\boldsymbol{x}_1, \boldsymbol{x}_2))$ is also a kernel.

Problem 8: Probabilistic inference & SVD (Version A) (4 credits)

Consider a generative model where $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ is the observed variable and $\boldsymbol{a} \in \mathbb{R}^{N}$, $\boldsymbol{b} \in \mathbb{R}^{D}$ are the model parameters. We assume the following generative process:

$$p(\boldsymbol{X}|\boldsymbol{a}, \boldsymbol{b}) = \prod_{i=1}^{N} \prod_{j=1}^{D} p(X_{ij}|\boldsymbol{a}, \boldsymbol{b}) = \prod_{i=1}^{N} \prod_{j=1}^{D} \mathcal{N}(X_{ij}|a_i \cdot b_j, 1).$$

Here $\mathcal{N}(x|\mu, \sigma)$ denotes the density of the normal distribution

$$\mathcal{N}(x|\mu, \sigma) \propto \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

a) Suppose you observed $\boldsymbol{X}$. Derive a maximum likelihood estimate (MLE) $\boldsymbol{a}^\star, \boldsymbol{b}^\star$ of the parameters $\boldsymbol{a}$, $\boldsymbol{b}$.

Hint: SVD can be helpful here. No need to take derivatives.

☐ 0
☐ 1
☐ 2

b) Suppose all singular values of the observed matrix $\boldsymbol{X}$ are distinct. Is the MLE of the parameters $\boldsymbol{a}$, $\boldsymbol{b}$ unique in this case? Justify your answer.

☐ 0
☐ 1
☐ 2

Problem 8: Probabilistic inference & SVD (Version B) (4 credits)

Consider a generative model where $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ is the observed variable and $\boldsymbol{a} \in \mathbb{R}^{N}$, $\boldsymbol{b} \in \mathbb{R}^{D}$ are the model parameters. We assume the following generative process:

$$p(\boldsymbol{X}|\boldsymbol{a}, \boldsymbol{b}) = \prod_{i=1}^{N} \prod_{j=1}^{D} p(X_{ij}|\boldsymbol{a}, \boldsymbol{b}) = \prod_{i=1}^{N} \prod_{j=1}^{D} \mathcal{N}(X_{ij}|a_i \cdot b_j, 1).$$

Here $\mathcal{N}(x|\mu, \sigma)$ denotes the density of the normal distribution

$$\mathcal{N}(x|\mu, \sigma) \propto \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

0 1 2

a) Suppose you observed $\boldsymbol{X}$. Derive a maximum likelihood estimate (MLE) $\boldsymbol{a}^\star, \boldsymbol{b}^\star$ of the parameters $\boldsymbol{a}$, $\boldsymbol{b}$.

Hint: SVD can be helpful here. No need to take derivatives.

0 1 2

b) Suppose all singular values of the observed matrix $\boldsymbol{X}$ are distinct. Is the MLE of the parameters $\boldsymbol{a}$, $\boldsymbol{b}$ unique in this case? Justify your answer.

Problem 9: Dimensionality Reduction (Version A) (2 credits)

Figure 24.1 shows a scatter plot of your two-dimensional data ($N = 13$ instances). You want to apply a non-linear dimensionality reduction technique based on neighbor graphs (e.g. T-SNE or UMAP). As a first step you compute the $N \times N$, weighted adjacency matrix representing the neighbor graph. Assume that the weights are computed with

$$p_{j|i} = \frac{\exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2\right)}{\sum_{k \neq i} \exp\left(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma^2\right)}$$

where $\boldsymbol{x}_i \in \mathbb{R}^2$ and you set $p_{i|i} = 0$. Finally, you obtain the similarity between instances i and j with $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$.

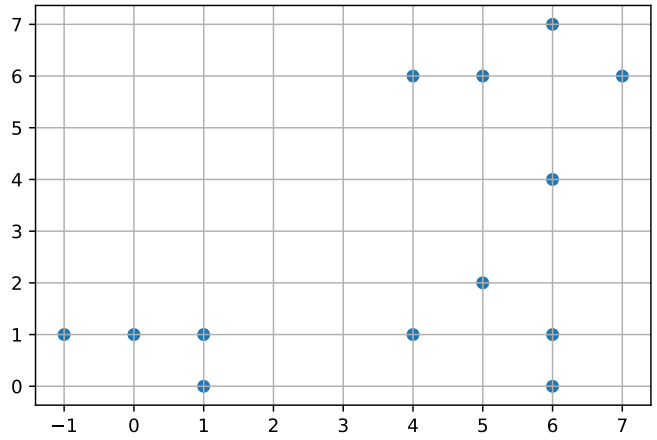
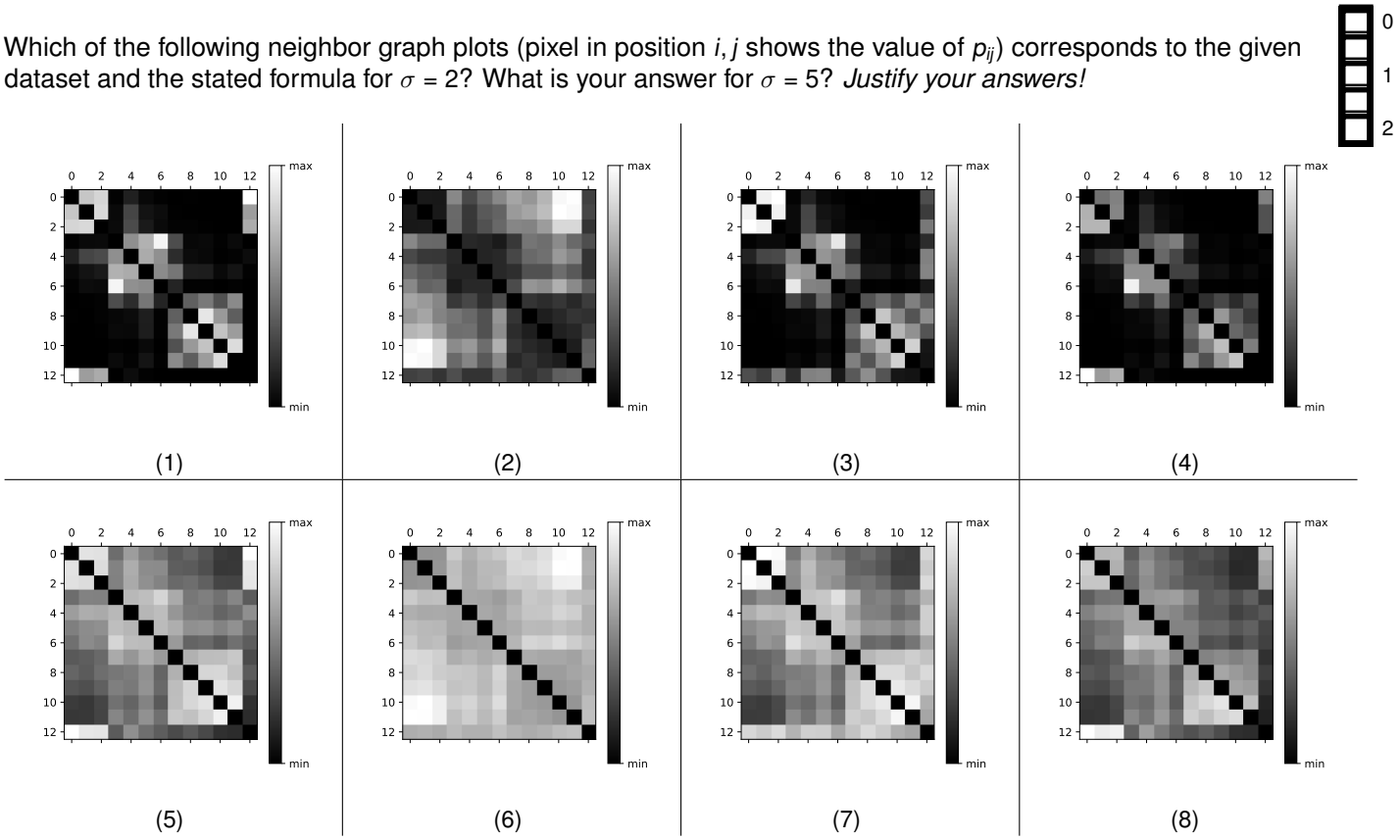


Figure 24.1: Scatter plot of the data

Which of the following neighbor graph plots (pixel in position i, j shows the value of p_{ij}) corresponds to the given dataset and the stated formula for $\sigma = 2$? What is your answer for $\sigma = 5$? *Justify your answers!*



Problem 9: Dimensionality Reduction (Version B) (2 credits)

Figure 25.1 shows a scatter plot of your two-dimensional data ($N = 13$ instances). You want to apply a non-linear dimensionality reduction technique based on neighbor graphs (e.g. T-SNE or UMAP). As a first step you compute the $N \times N$, weighted adjacency matrix representing the neighbor graph. Assume that the weights are computed with

$$p_{j|i} = \frac{\exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / 2\sigma^2\right)}{\sum_{k \neq i} \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_k\right\|^2 / 2\sigma^2\right)}$$

where $\boldsymbol{x}_i \in \mathbb{R}^2$ and you set $p_{i|i} = 0$. Finally, you obtain the similarity between instances i and j with $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$.

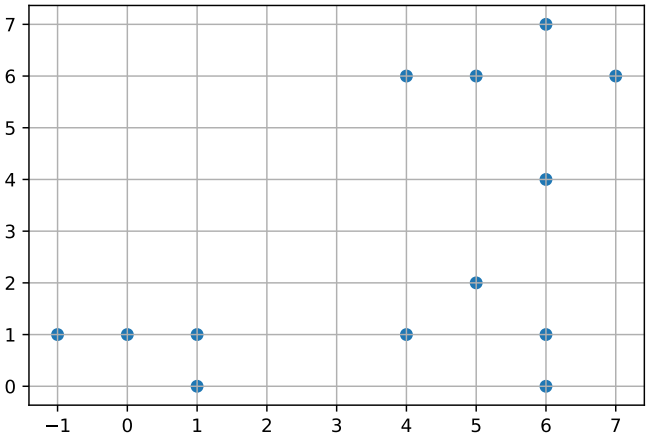
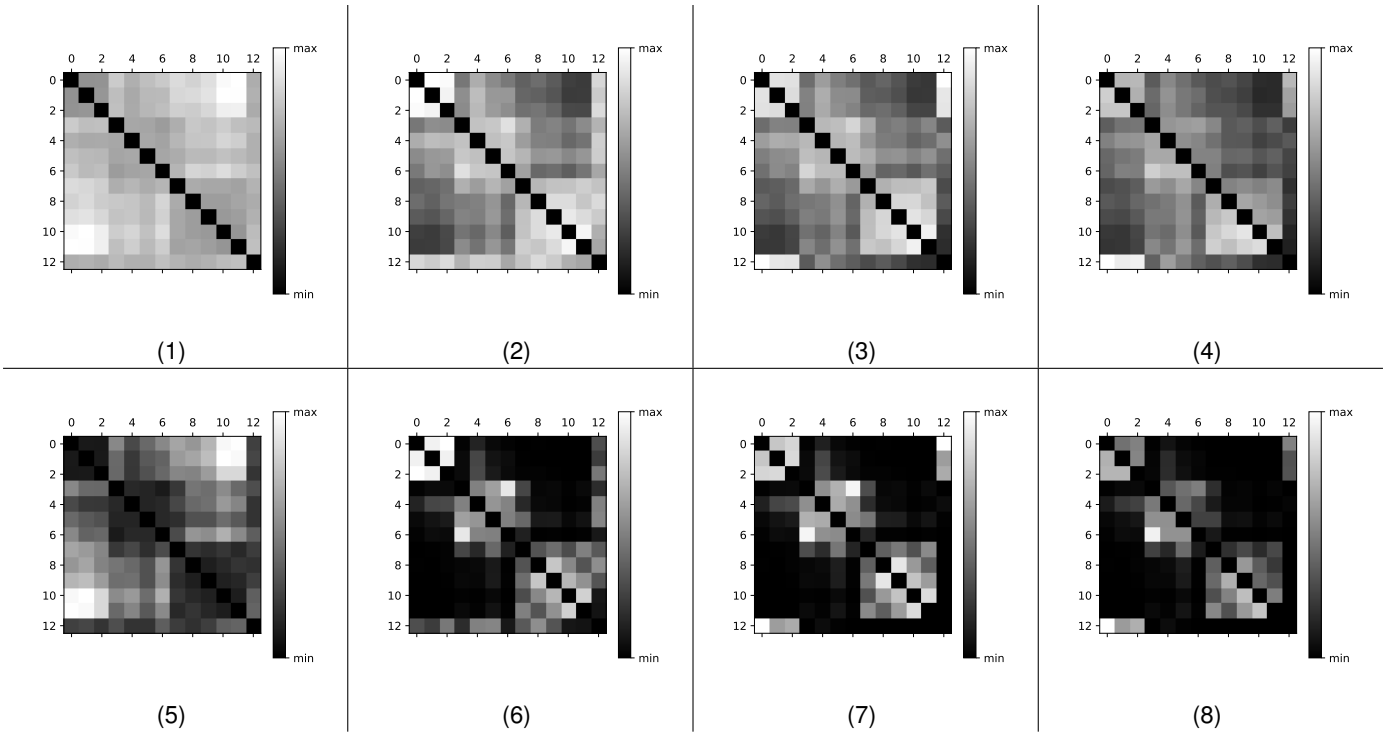


Figure 25.1: Scatter plot of the data

0 ☐
1 ☐
2 ☐

Which of the following neighbor graph plots (pixel in position i, j shows the value of p_{ij}) corresponds to the given dataset and the stated formula for $\sigma = 2$? What is your answer for $\sigma = 5$? *Justify your answers!*



Problem 9: Dimensionality Reduction (Version C) (2 credits)

Figure 26.1 shows a scatter plot of your two-dimensional data ($N = 13$ instances). You want to apply a non-linear dimensionality reduction technique based on neighbor graphs (e.g. T-SNE or UMAP). As a first step you compute the $N \times N$, weighted adjacency matrix representing the neighbor graph. Assume that the weights are computed with

$$p_{j|i} = \frac{\exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / 2\sigma^2\right)}{\sum_{k \neq i} \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_k\right\|^2 / 2\sigma^2\right)}$$

where $\boldsymbol{x}_i \in \mathbb{R}^2$ and you set $p_{i|i} = 0$. Finally, you obtain the similarity between instances i and j with $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$.

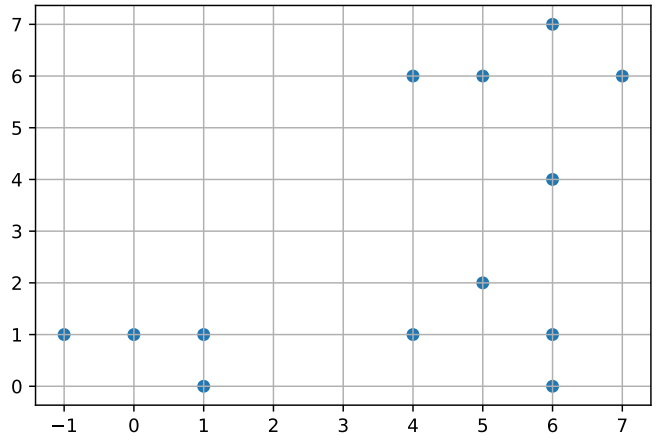
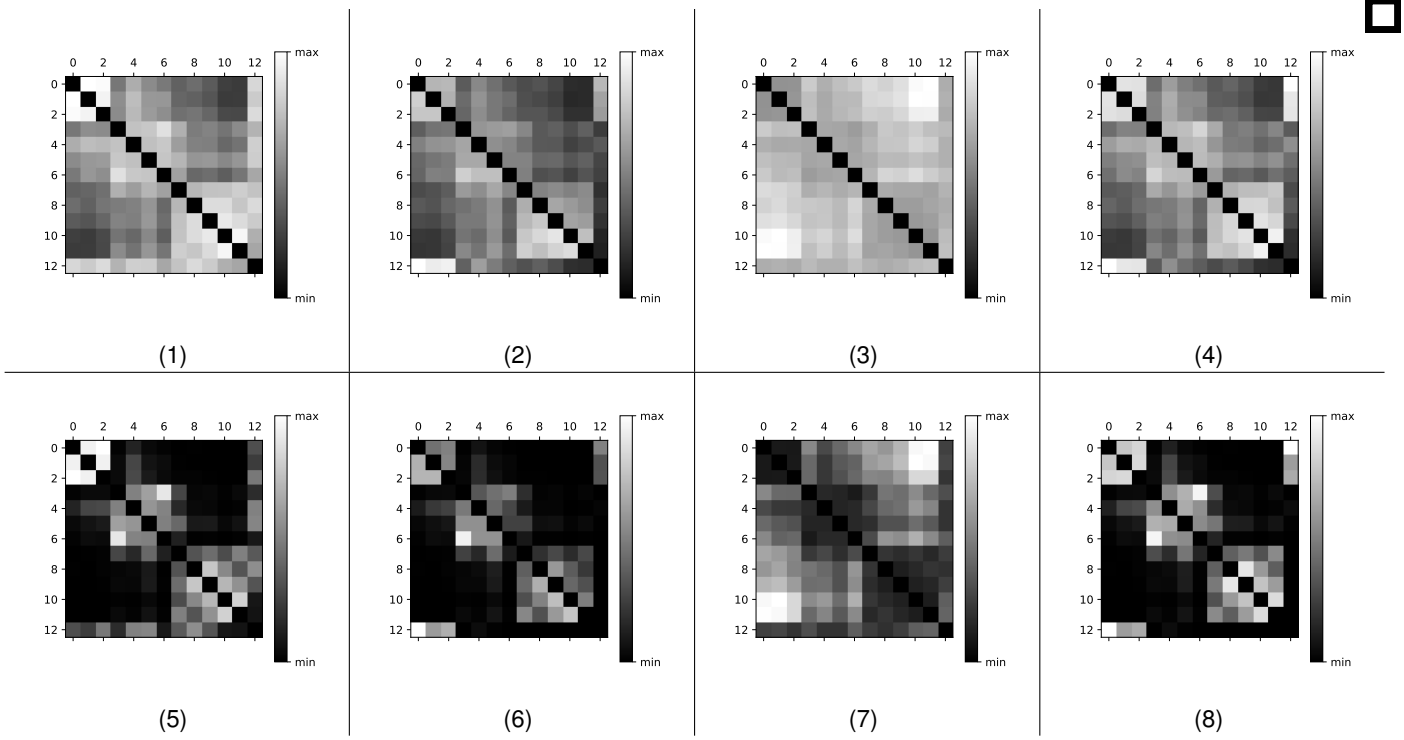


Figure 26.1: Scatter plot of the data

Which of the following neighbor graph plots (pixel in position i, j shows the value of p_{ij}) corresponds to the given dataset and the stated formula for $\sigma = 2$? What is your answer for $\sigma = 5$? *Justify your answers!*



Problem 9: Dimensionality Reduction (Version D) (2 credits)

Figure 27.1 shows a scatter plot of your two-dimensional data ($N = 13$ instances). You want to apply a non-linear dimensionality reduction technique based on neighbor graphs (e.g. T-SNE or UMAP). As a first step you compute the $N \times N$, weighted adjacency matrix representing the neighbor graph. Assume that the weights are computed with

$$p_{j|i} = \frac{\exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / 2\sigma^2\right)}{\sum_{k \neq i} \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_k\right\|^2 / 2\sigma^2\right)}$$

where $\boldsymbol{x}_i \in \mathbb{R}^2$ and you set $p_{i|i} = 0$. Finally, you obtain the similarity between instances i and j with $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$.

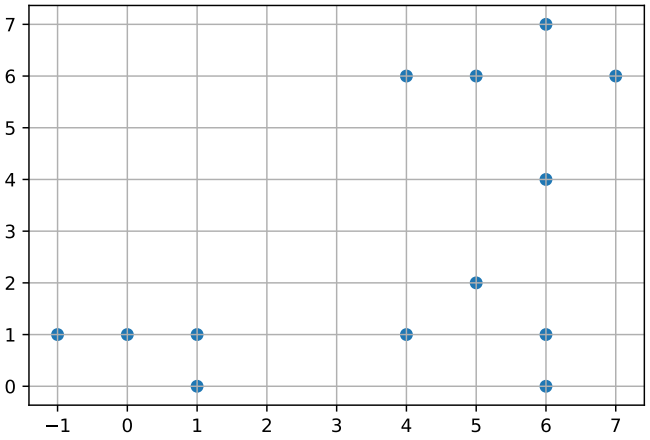
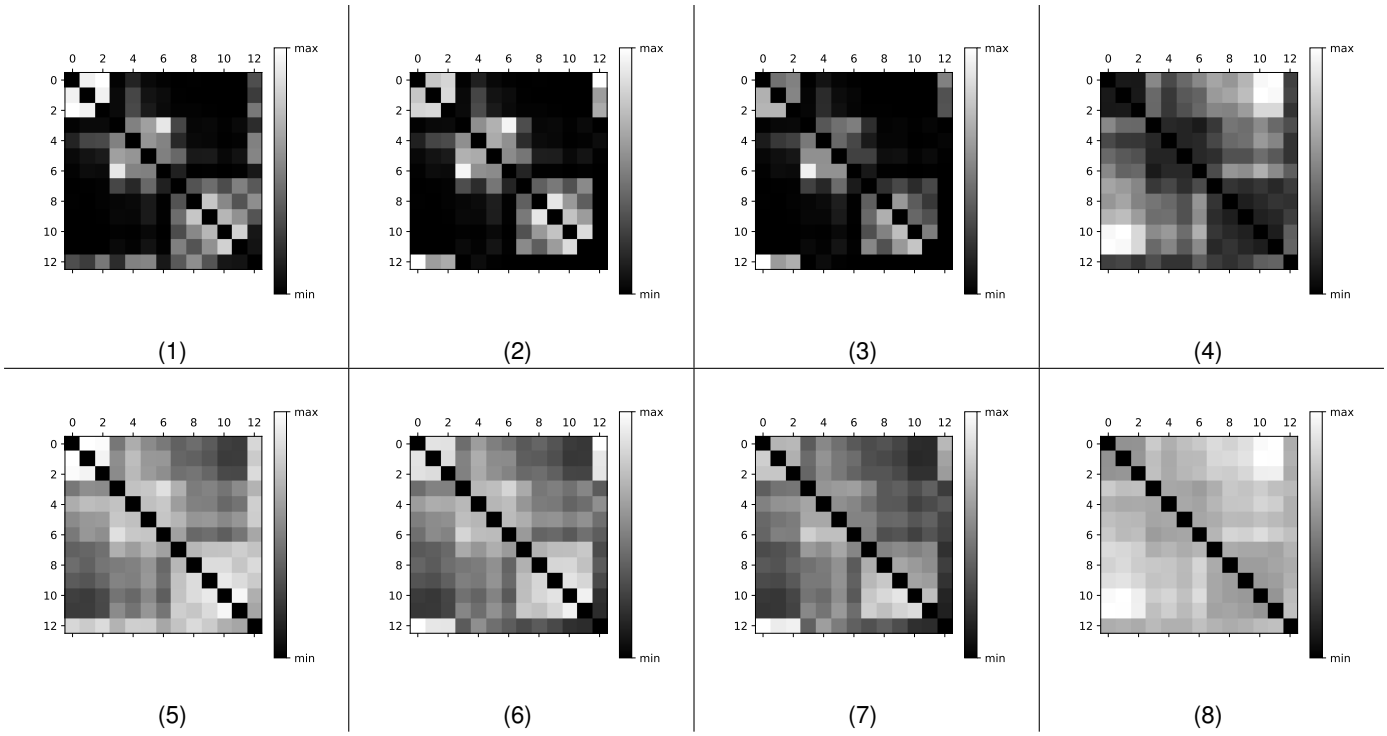


Figure 27.1: Scatter plot of the data

0 ☐
1 ☐
2 ☐

Which of the following neighbor graph plots (pixel in position i, j shows the value of p_{ij}) corresponds to the given dataset and the stated formula for $\sigma = 2$? What is your answer for $\sigma = 5$? *Justify your answers!*



Problem 10: Clustering (Version A) (4 credits)

a) Consider the K-means algorithm with two clusters and fixed centroids μ_1, μ_2. Prove that the decision boundary is a hyperplane.

Hint: Consider the equation that defines the decision boundary.

0 1 2

b) Now consider the Gaussian mixture model (GMM) with two clusters. In this case the assigned cluster label for instance $\boldsymbol{x}_i$ is given by

$$\arg\max_{k\in\{1,2\}} \gamma(\boldsymbol{z}_{ik}). \tag{1}$$

where $\gamma(\boldsymbol{z}_{ik})$ are the responsibilities. Is the decision boundary in a GMM with two clusters and fixed parameters π_k, $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ linear in general? If yes, give the associated hyperplane. If no, specify the conditions on the parameters such that the decision boundary is linear *if and only if* the conditions hold. Justify your answer.

0 1 2

a)

At the decision boundary we have

$$\|x-\mu_1\|_2 = \|x-\mu_2\|_2$$

$$\Leftrightarrow \|x-\mu_1\|_2^2 = \|x-\mu_2\|_2^2$$

$$\Leftrightarrow (x-\mu_1)^T(x-\mu_1) = (x-\mu_2)^T(x-\mu_2)$$

$$\Leftrightarrow x^2 - 2\mu_1^T x + \mu_1^2 = x^2 - 2\mu_2^T x + \mu_2^2$$

$$\Leftrightarrow \underbrace{\mu_1^2 - \mu_2^2}_{x_0} + \underbrace{(2\mu_2 - 2\mu_1)^T}_{w} x = 0$$