Cognitive Algorithms Test Exam

WS 2023/24

Please fill in below your full name, your matriculation number and field of studies. I hereby confirm that I feel capable to participate in this exam.

_____

Signature

Name	
Field of study	
Matriculation number	
Registration	☐ Via Moses ☐ Other

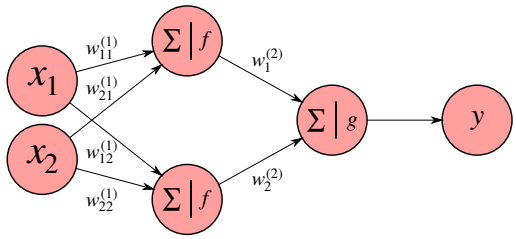
Section	Points	Score
Overview questions	14	
Linear Classification	25	
Kernel methods and Kernel Ridge Regression	7	
Unsupervised Learning	13	
MLP	14	
Cross-Validation	8	
Total	81	

1. Overview questions [14 Points total]

1.1 [1 Point] What do we mean when we say that our machine learning model generalizes well? (1-2 sentences)

Solution: A model generalizes well when its error on new/unseen data is small.

1.2 [2 Points] The neural network displayed below is _____ training algorithm with ___ hidden layer(s). Choose one Answer from the first column to fill in the first gap and one answer from the second column to fill in the second gap.



- ◯ a supervised
- ◯ an unsupervised
- ◯ 1
- ◯ 2

Solution: supervised, 1

1.3 [1 Point] The neural network as displayed in the question above is a linear method.

Hint! When we talk about linear methods (including linear regression), we mean methods that are linear in $\boldsymbol{w}$, i.e. $f(x) = \boldsymbol{w}^\top x$ where $\boldsymbol{w}$ can also contain a non-linear transformation of the data and the offset/bias β.

- ◯ True
- ◯ False
- ◯ depends on f
- ◯ depends on g
- ◯ depends on f and g

Solution: depends on f

1.4 [2 Points] Given a fixed regularization parameter $\lambda > 0$ ridge regression...

... always has a lower training error than linear regression.

- ◯ True
- ◯ False

... always has a lower test error than linear regression.

- ◯ True
- ◯ False

Solution: False, False

1.5 [1.5 Points] Assume the covariance between observations X and their labels Y is less than 0. Which statement(s) are always true?

- ◯ The regression function given by ordinary least squares will have negative slope.
- ◯ The correlation between X and Y is -1.
- ◯ Either X or Y has negative variance.

Solution: The regression function given by ordinary least squares will have negative slope.

1.6 [1.5 Points] Which statement(s) are true for kernel methods (as presented in this course)?

- ◯ You need to store all the training data to predict labels of new data points.
- ◯ For every valid kernel function k, there is a feature map ϕ, such that $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$.
- ◯ Kernel ridge regression gives better results and is faster to compute than ridge regression.

Solution: You need to store all the training data, and there is a feature map ϕ as specified.

1.7 [3 Points] Name 3 algorithms we discussed in the lecture that can be used to address classification problems.

Solution: LDA, NCC, Perceptron, MLP

1.8 [1 Point] Name one situation in which it makes sense to use a kernelized algorithm? (1–2 sentences)

Solution:

- When we want to use a mapping that is expensive/impossible to compute
- Linear Kernel Methods make sense if $d \gg n$

1.9 [1 Point] "When the amount of data increases and the algorithm and hyperparameters stay the same, overfitting is more likely." True or false? Explain in one or two sentences.

Solution:

- This is a false statement
- With more data, we introduce more constraints on the function we try to model, making it less likely to overfit

2. Linear Classification [25 Points total]

2.1 [8 Points] Recall that the error function of the Perceptron was given by:

$$\mathcal{E}(\boldsymbol{w}) = -\sum_{m \in \mathcal{M}} \boldsymbol{w}^\top \boldsymbol{x}_m y_m \tag{1}$$

Briefly define, in words, $\boldsymbol{w}$, $\boldsymbol{x}_m$, y_m, and $\mathcal{M}$. **[2 Points]**

Solution:

$\boldsymbol{w}$ is the weight vector that the Perceptron learns.

$\boldsymbol{x}_m$ are the data points in the training set, $\boldsymbol{x}_m \in \mathbb{R}^d$.

y_m are the labels, $y_m \in \{-1, +1\}$.

$\mathcal{M}$ is the index set of misclassified data points.

Why do we include a minus sign in front of the summation? **[2 Points]**

Solution: Because $\boldsymbol{w}^\top \boldsymbol{x}_m$ is the opposite sign of y_m for misclassified data points. Therefore, $\boldsymbol{w}^\top \boldsymbol{x}_m y_m$ is always negative for misclassified data points. [1 Point] We want our error function to be positive and large when we have a lot of misclassified data points and at a minimum of zero for no misclassified data points. [1 Point].

Assume we do not want to determine $\mathcal{M}$ before calculating the error of $\boldsymbol{w}$. Therefore, we rewrite the error function as $\mathcal{E}_{new}(\boldsymbol{w}) = \sum_{i \in \mathcal{X}} \max(0, -\boldsymbol{w}^\top \boldsymbol{x}_i y_i)$, where $\mathcal{X}$ is the index set of all data points. Show that $\mathcal{E}_{new}$ is equivalent to the function $\mathcal{E}$ given as equation (1) above. **[4 Points]**.

Solution:

- if x_i is correctly classified, $-\boldsymbol{w}^\top \boldsymbol{x}_i y_i < 0$ and $\max(0, -\boldsymbol{w}^\top \boldsymbol{x}_i y_i) = 0$, i.e., correctly classified data points do not contribute to the error
- if x_i is wrongly classified, $\max(0, -\boldsymbol{w}^\top \boldsymbol{x}_i y_i) = -\boldsymbol{w}^\top \boldsymbol{x}_i y_i$

2.2 [3 Points] Again recall the error of the Perceptron (see equation (1)). Assuming that we use the augmented notation, i.e., that $\boldsymbol{w}^\top \boldsymbol{x}_m = \begin{bmatrix}\beta & w_1 & \dots & w_d\end{bmatrix} \begin{bmatrix}-1 \\ x_{m1} \\ \dots \\ x_{md}\end{bmatrix}$, calculate the update that is applied to β in each iteration of Stochastic Gradient Descent (SGD) during the training of the Perceptron.

Hint! SGD uses the error of a single, randomly chosen misclassified data point, given by $\mathcal{E}_{xm}(\boldsymbol{w}) = -\boldsymbol{w}^\top \boldsymbol{x}_m y_m$

Solution: $\frac{\partial \mathcal{E}_{xm}(\boldsymbol{w})}{\partial \beta} = y_m$
$\beta^{\tau+1} \leftarrow \beta^\tau - \eta y_m$

2.3 [5 Points] You are given the following dataset consisting of two **infinite** classes C_{+1} (with all labels being $+1$) and C_{-1} (with all labels being -1):

$$C_{+1} := \left\{ \begin{bmatrix} x \\ x \end{bmatrix} \middle| x \in \mathbb{R}, x > 0 \right\}$$
$$C_{-1} := \left\{ \begin{bmatrix} x \\ x+1 \end{bmatrix} \middle| x \in \mathbb{R}, x > 0 \right\}$$
$$X := C_{+1} \cup C_{-1} \qquad \text{(i.e., } X \text{ is the union of } C_{+1} \text{ and } C_{-1}\text{)}$$

Which algorithm has a chance of calculating a solution in a finite amount of time? **[1 Point]**

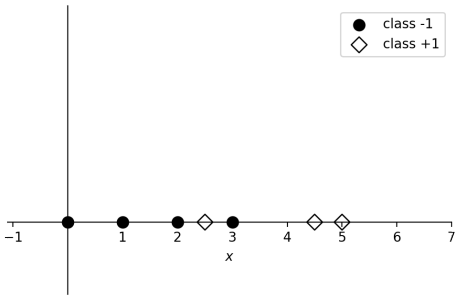
- ◯ Perceptron using Gradient Descent
- ◯ Perceptron using Stochastic Gradient Descent
- ◯ LDA
- ◯ None of the above

Solution: Perceptron using Stochastic Gradient Descent.

Why did you choose your answer? **[4 Points]**

Solution: The update step of Gradient Descent is impossible to calculate within a finite amount of time, since we have infinitely many data points. A similar reason excludes LDA. The update step of Stochastic Gradient Descent can be calculated in a finite amount of time.

2.4 [9 Points] Consider the following 1-dimensional dataset $X = \begin{bmatrix} 0 & 1 & 2 & 2.5 & 3 & 4.5 & 5 \end{bmatrix}$, with its corresponding labels $y = \begin{bmatrix} -1 & -1 & -1 & +1 & -1 & +1 & +1 \end{bmatrix}$. It is also illustrated in the plot below:



Consider the nearest centroid classifier (NCC). Write down the formulas for $\boldsymbol{w}$ and β that correspond to the NCC **[2 Points]**. Calculate $\boldsymbol{w}$ and β for the given dataset. **[2 Point]**. Calculate the **training accuracy** that NCC would achieve for this dataset **[1 Point]**.

Solution: $w = w_{+1} - w_{-1} = 4 - 1.5 = 2.5$, $\beta = \frac{1}{2}(w_{+1}^2 - w_{-1}^2) = \frac{1}{2}w(w_{+1} + w_{-1}) = 6.875$
all data points except x_4 and x_5 correctly classified, therefore the train accuracy is: $\frac{5}{7}$

What is the highest accuracy that a linear discriminator can achieve on this dataset? **[1 Point]**.

Denote by n_+ the number of training data points in class $+1$, and n_- in class -1. Now consider Fisher's linear discriminant analysis (LDA). In the lecture, we derived a β for LDA that depended on n_+ and n_-. What problem is addressed by this dependence? You consider a new training set where n_- is larger than in the first training set, and n_+ is the same. Should β increase, decrease, or stay the same, with respect to the first training set? **[3 Points]**.

Solution: The highest accuracy is 6/7. Having β depend on $n+$ and $n-$ reflects the class imbalance in the training set.

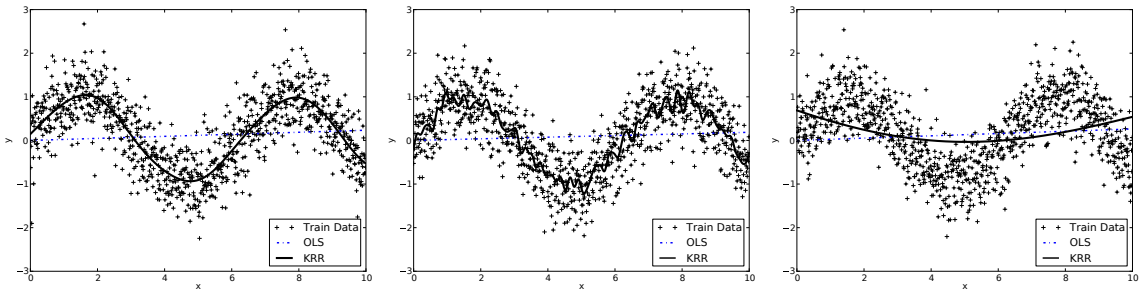
If n_- increases and n_+ does not change, then β should increase (intuition: because a test data point x is classified as class -1 if $\boldsymbol{w}^\top \boldsymbol{x} < \beta$, this decision should become more likely when the proportion of class -1 is higher).

3. Kernel methods and Kernel Ridge Regression [7 Points total]

3.1 [4 Points]

1. **[3 points]** We used a Kernel Ridge Regression with a Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right)$ on training data that follows a sine-function. Below you find the results for three different kernel widths. Indicate which of the following three labels corresponds to which of the plots:

   $\sigma = 10,$ $\sigma = 1,$ $\sigma = 0.1$



Solution: In order of the plots 1, 0.1, 10.

2. **[1 point]** Explain intuitively how the kernel width σ affects the learned model.

Solution: When σ is small, $k(\boldsymbol{x}, \boldsymbol{y})$ rapidly decays to 0 when $\|\boldsymbol{x} - \boldsymbol{y}\|$ increases. Thus, only training data in a close neighborhood of a test data-point influence the prediction. When σ is large, the opposite happens.

3.2 [3 Points] You are given the following feature map

$$\phi(x)^\top = (x_1^2, \sqrt{2}x_1x_2, x_2^2), \text{ where } x \in \mathbb{R}^2.$$

Show that the dot product defines a kernel function, i.e. show that

$$\phi(x)^\top \phi(y) = k(x, y) = \left(x^\top y\right)^2$$

where again $x, y \in \mathbb{R}^2$.

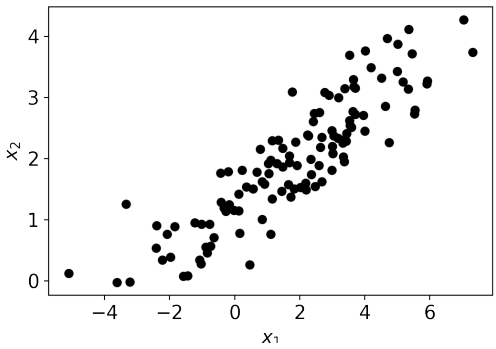
Solution:

$$\begin{aligned}
\phi(x)^\top \phi(y) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(y_1^2, \sqrt{2}y_1y_2, y_2^2)^\top \\
&= x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2 \\
&= (x_1y_1 + x_2y_2)^2 \\
&= ((x_1, y_1)(x_1, y_1)^\top)^2 \\
&= (x^\top y)^2 = k(x, y)
\end{aligned}$$

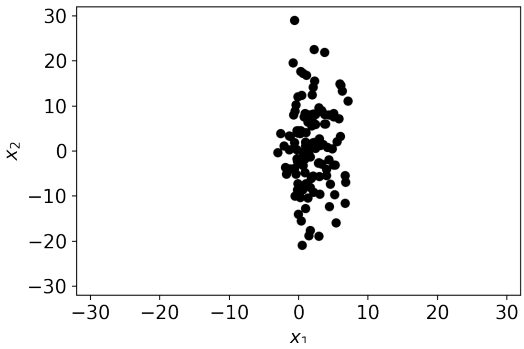
We stated in the course that $x^\top y$ defines a kernel function, and that if $k(x, y)$ is a kernel function, then $(k(x, y))^2$ is a kernel function as well.

4. Unsupervised Learning [13 Points total]

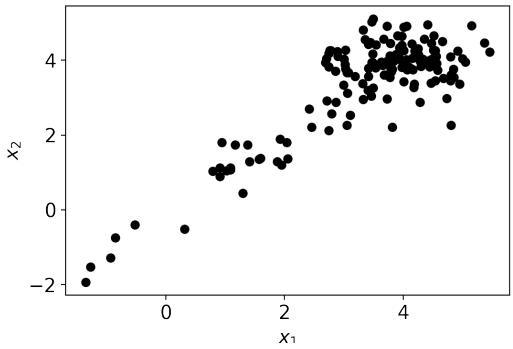
4.1 [7 Points] Below you see three plots of data sets (called "A", "B", and "C").



Data Set "A"



Data Set "B"



Data Set "C"

Which one is most likely not stemming from a Gaussian distribution? **[1 Point]**

Solution: Data Set "C"

For two of the datasets above, draw plausible first and second principal components into their corresponding plots **[2 Points]**. Also give a **plausible** ratio $\frac{\lambda_1}{\lambda_2}$ for each of the two datasets, where λ_1 and λ_2 are the eigenvalues of the covariance matrix **[2 Points]**.

Dataset 1: $\frac{\lambda_1}{\lambda_2} =$

Dataset 2: $\frac{\lambda_1}{\lambda_2} =$

Solution: Dataset "A": $\frac{\lambda_1}{\lambda_2} = 23$
Dataset "B": $\frac{\lambda_1}{\lambda_2} = 33$
Dataset "C": $\frac{\lambda_1}{\lambda_2} = 18$

In all these cases, a number x with $1 < x < 1000$ would be accepted.

You want to calculate the variance of each dataset along each dimension. For that you wrote your own python function `var`, which measures the average difference of a single dimension of a dataset to its corresponding mean:

```
def var(X : np.ndarray) -> float:
    variance = 0
    mean = np.mean(X)
    n_datapoints = X.shape[0]
    for data_index in range(n_datapoints):
        variance += X[data_index] - mean
    return variance/n_datapoints
```

You have called this function on each dimension of the datasets "B" and "C" and stored them in vectors, where each row corresponds to the estimated variance of that dimension. Unfortunately you do not get very plausible results:

Variance of data set "B": $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Variance of data set "C": $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Give a plausible output of the function on dataset "A" [**1 Points**]:

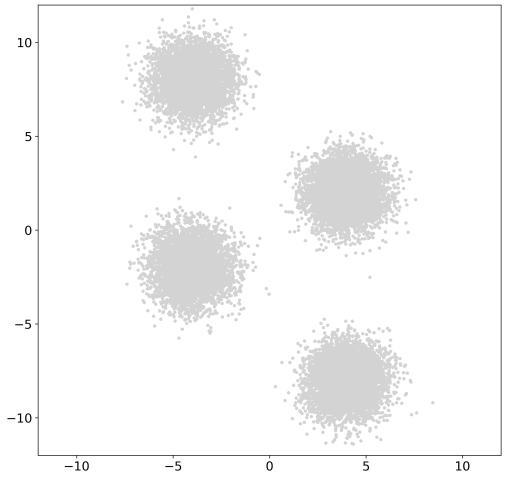
Solution: Variance of data set "A": $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

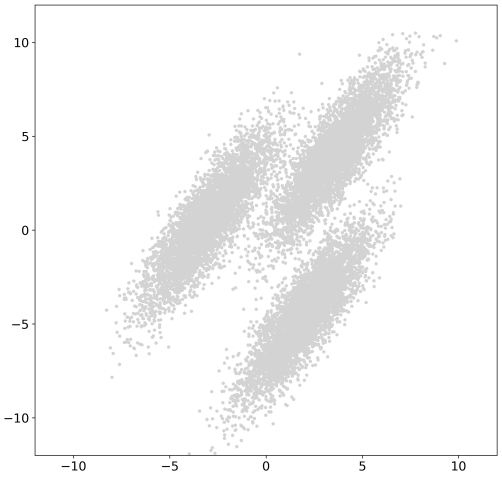
Why does your function `var` give these wrong results? [**2 Points**]

Solution: You forgot to include a square around the difference between the data points and the mean.

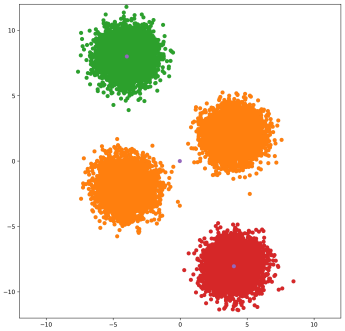
4.2 [6 Points]

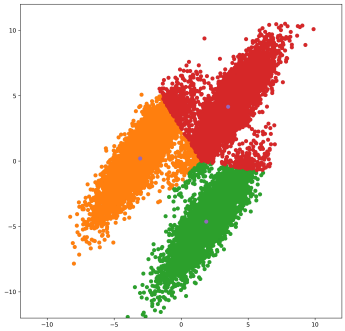
1. **[4 points]** In the Figure below you find two exemplary unlabeled data sets. Draw for each a plausible result of the k-means algorithm after 50 iterations, i.e. mark plausible **clusters** and **cluster centers** found by k-means for $k = 3$. Initial cluster centers were randomly drawn from the set of data points.





Solution:



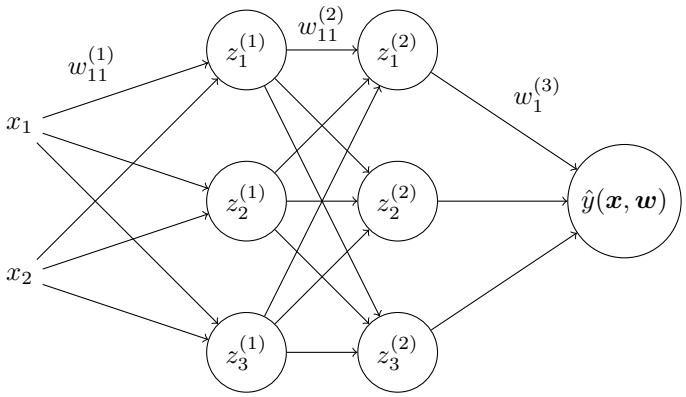


2. **[2 points]** For your bachelor thesis, you apply a classification algorithm to very high dimensional data you have recorded. Your supervisor is concerned that your features are still very correlated and suggests applying an unsupervised algorithm to reduce the dimensionality of your data before applying the classification algorithm. What does she mean by that? Explain briefly in 1–2 sentences and state one possible algorithm which we discussed in the lecture.

Solution: Correlation between features can have negative effects on classification algorithms (in the lecture, we explicitly saw this with the NCC). A possible solution is to use PCA: after applying PCA, the features are uncorrelated. In this case, the dimensionality can be reduced by retaining only the principal components with the largest eigenvalues and disregarding the rest.

5. MLP [14 Points total]

5.1 [10 Points] Consider the following MLP architecture:



In this MLP, $z_1^{(1)}$ is calculated as

$$z_1^{(1)} = f(a_1^{(1)}) = f(w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2),$$

where f is a non-linear activation function used to calculate all activations. The other $z_i^{(j)}$ are defined analogously, and $\hat{y}$ does not use any activation function.

For an input $\boldsymbol{x}$ with label y, let

$$\mathcal{E}(\boldsymbol{x}, \boldsymbol{w}, y) := \frac{1}{2} \left(\hat{y}(\boldsymbol{x}, \boldsymbol{w}) - y\right)^2$$

be the error function of the MLP.

- **[2 points]** You would like to update the weights in the MLP to decrease its error. Given a learning rate $\eta > 0$, which of the following examples of weight updates are correct?

  ◯ $w_{11}^{(1)} \leftarrow w_{11}^{(1)} - \eta \frac{\partial \mathcal{E}}{\partial w_{11}^{(1)}}$
  ◯ $w_{11}^{(1)} \leftarrow w_{11}^{(1)} - \eta \frac{\partial \mathcal{E}}{\partial y}$
  ◯ None of the above.

  Solution: Only the first one is correct.

- **[8 points]** You would like to update the weights $w_1^{(3)}$ and $w_{11}^{(2)}$. Calculate the weight updates, i.e. derive $\mathcal{E}$ with respect to both these weights. Carry out all partial derivatives iteratively. For the derivative of f, you can simply write f'.

  Hint! Chain rule for nested functions:

  $$\frac{\partial}{\partial x_j} f(g_1(\boldsymbol{x}), \ldots, g_n(\boldsymbol{x})) = \sum_{i=1}^{n} \left(\frac{\partial}{\partial g_i} f(g_1, \ldots, g_n)\right) \cdot \left(\frac{\partial}{\partial x_j} g_i(\boldsymbol{x})\right).$$

Solution:

–

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_1^{(3)}} &= \frac{\partial}{\partial w_1^{(3)}} \frac{1}{2} (\hat{y} - y)^2 \\ &= (\hat{y} - y) \frac{\partial}{\partial w_1^{(3)}} \hat{y} \\ &= (\hat{y} - y) \frac{\partial}{\partial w_1^{(3)}} \left(w_1^{(3)} z_1^{(2)} + w_2^{(3)} z_2^{(2)} + w_3^{(3)} z_3^{(2)} \right) \\ &= (\hat{y} - y) z_1^{(2)}.\end{aligned}$$

–

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{11}^{(2)}} &= \frac{\partial}{\partial w_{11}^{(2)}} \frac{1}{2} (\hat{y} - y)^2 \\ &= (\hat{y} - y) \frac{\partial}{\partial w_{11}^{(2)}} \hat{y} \\ &= (\hat{y} - y) \frac{\partial}{\partial w_{11}^{(2)}} \left(w_1^{(3)} z_1^{(2)} + w_2^{(3)} z_2^{(2)} + w_3^{(3)} z_3^{(2)} \right) \\ &= (\hat{y} - y) w_1^{(3)} \frac{\partial}{\partial w_{11}^{(2)}} z_1^{(2)} \\ &= (\hat{y} - y) w_1^{(3)} \frac{\partial}{\partial w_{11}^{(2)}} f(a_1^{(2)}) \\ &= (\hat{y} - y) w_1^{(3)} f'(a_1^{(2)}) \frac{\partial}{\partial w_{11}^{(2)}} a_1^{(2)} \\ &= (\hat{y} - y) w_1^{(3)} f'(a_1^{(2)}) \frac{\partial}{\partial w_{11}^{(2)}} \left(w_{11}^{(2)} z_1^{(1)} + w_{21}^{(2)} z_2^{(1)} + w_{31}^{(2)} z_3^{(1)} \right) \\ &= (\hat{y} - y) w_1^{(3)} f'(a_1^{(2)}) z_1^{(1)}.\end{aligned}$$

5.2 [4 Points] Consider the same MLP as above. In this subtask, we will be using the tanh activation function, i.e., $f(z_i^{(k)}) = \tanh(z_i^{(k)})$. Further imagine that we have determined well-working values for all of the weights of the network.
Describe alternative weights $w'^{(2)}_{ij}$ and $w'^{(3)}_{i}$ that are different to $w^{(2)}_{ij}$ and $w^{(3)}_{i}$ but lead to the **exact same** overall network output.

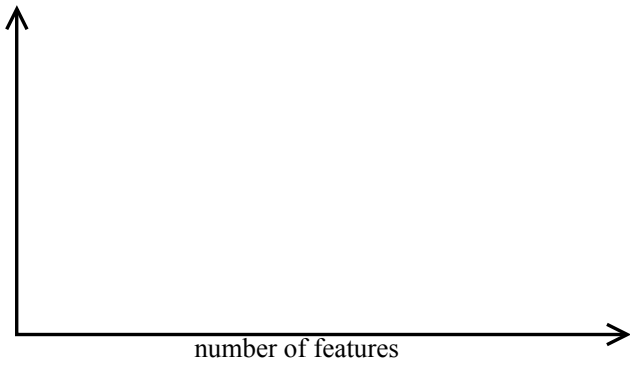
Hint! Consider the fact that tanh *is an odd function, i.e., that* $\tanh(-a) = -\tanh(a)$

Solution: Multiply each weight $w^{(2)}_{ij}$ and $w^{(3)}_{i}$ by -1, i.e., $w'^{(2)}_{ij} \leftarrow -w^{(2)}_{ij}$ and $w'^{(3)}_{i} \leftarrow -w^{(3)}_{i}$.
Alternatively you can describe a permutation of the weights.

6. Cross-Validation [8 Points total]

6.1 [3 Points] Suppose you model the non-linear relationship between a one-dimensional input x and a one-dimensional output y as an mth order polynomial, i.e. $y = w_0 + w_1 x + w_2 x^2 + \ldots + w_m x^m$. The number of training points is fixed, and you estimate the parameters $w_0, w_1, \ldots, w_m$ by linear regression.

Draw a graph showing two curves: training error vs. the number of features m and cross-validation error vs. the number of features m, annotate both curves. Include cases where over- and under-fitting occur and mark them in the plot.



Solution:

As a function of m, the training error starts high and decreases down to zero.

The cross-validation error starts high (under-fitting), decreases, but usually not to zero, and then increases again (over-fitting).

In any case, the cross-validation error is higher than the training error.

6.2 [2 Points] Find the bugs in the cross-validation algorithm below and correct them.

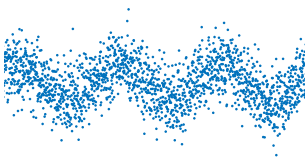
Algorithm 1 Cross-Validation

Require: Data $\{(x_1, y_1) \ldots, (x_N, y_N)\}$, Number of CV folds F

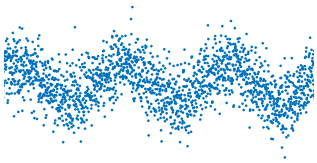
1: Split data in F overlapping folds
2: **for** Fold $f = 1, \ldots, F$ **do**
3: Train model on folds $\{1, \ldots, F\}$
4: Compute prediction on fold f
5: **end for**
6: **return** average prediction error

Solution: The folds must not overlap. The model has to be trained on folds $\{1, \ldots, F\} \setminus f$. (Alternatively, another error is that we need to compute the prediction ERROR.)

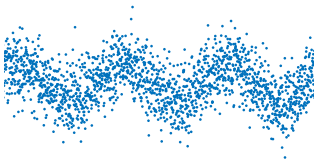
6.3 [3 Points] Below you find 3 equal plots with data points. Sketch possible solutions from a polynomial regression, one that under-fits (a), over-fits (b) and one good fit (c)



(a) under-fitting



(b) over-fitting



(c) good fit