

QUESTION 1

Since all necessary values seem to be given, we simply calculate the expected in-sample error using the formula given.

The smallest N that has an in-sample error of $E_{in} > 0.008$ is $N = 100$. The answer is c .

QUESTION 2

First, generate some points that fit the classification boundaries given in the figure. Create some random points with $-1 \leq x_1, x_2 \leq 1$. Then, classify them using the following conditions: $+1$ for $(\|\mathbf{x} - (1, 0)\| > 0.7 \wedge |x_1| < 0.8)$. Apply the nonlinear transform to the points and use the perceptron algorithm to learn the weight vector $\tilde{\mathbf{w}}$ for classifying them in \mathcal{Z} -space. Here, \tilde{w}_1 is negative while \tilde{w}_2 is positive, therefore answer d applies. The classified points in \mathcal{X} and \mathcal{Z} space are shown in figure 1.

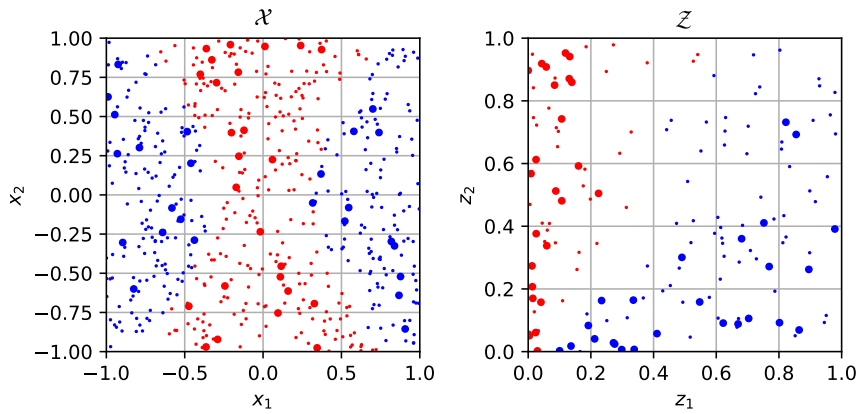


Fig. 1: Classified points in \mathcal{X} and \mathcal{Z} space. Large points are training points, small ones belong to the test data set.

QUESTION 3

Find the VC dimension for a 4th order nonlinear transform function. The VC-dimension d_{VC} for a nonlinear feature transform Φ_Q of order Q can be

$$d_{VC} \leq \frac{Q(Q+3)}{2} + 1. \quad (1)$$

See the book page 105 for more. For $Q = 4$ we get $d_{VC} \leq 15$. Thus, the smallest choice that is not smaller than 15 is $c: 15$.

QUESTION 4

Find $\frac{\partial E}{\partial u}$ where $E(u, v) = (ue^v - 2ve^{-u})^2$.

First, we apply the chain rule to get rid of the square brackets. Substitute the inner term by w so that

$$w = (ue^v - 2ve^{-u}). \quad (2)$$

We can now calculate the derivative of the outer term w^2 and the inner term separately and combine them to

$$\frac{\partial E}{\partial u} = \frac{\partial}{\partial w} w^2 \cdot \frac{\partial}{\partial u} (ue^v - 2ve^{-u}). \quad (3)$$

The outer derivative is given by

$$\frac{\partial}{\partial w} w^2 = 2w \quad (4)$$

$$= 2(ue^v - 2ve^{-u}) \quad (5)$$

The inner derivative is given by

$$\frac{\partial}{\partial u} (ue^v - 2ve^{-u}) = \frac{\partial}{\partial u} ue^v - \frac{\partial}{\partial u} 2ve^{-u} \quad (6)$$

$$= 1 \cdot e^v - 2v(-e^{-u}) \quad (7)$$

$$= e^v + 2ve^{-u} \quad (8)$$

We can now combine the inner and outer parts to get

$$\frac{\partial E}{\partial u} = 2(ue^v - 2ve^{-u})(e^v + 2ve^{-u}). \quad (9)$$

This corresponds to answer e .

QUESTION 5

Compute the minimum number of generations to optimize the vector $w = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ that lives in the (u, v) space until $E(w) < 10^{-14}$.

First, we need the partial derivative in v -direction as well. Using the process outlined in question 5 above, we get

$$\frac{\partial E}{\partial v} = 2(ue^v - 2ve^{-u})(ue^v - 2e^{-u}). \quad (10)$$

With both partial derivatives we can construct the gradient vector

$$\nabla E = \begin{pmatrix} \frac{\partial E}{\partial u} \\ \frac{\partial E}{\partial v} \end{pmatrix} \quad (11)$$

This vector lives in the (u, v) space and points into the direction of the largest change of E . To improve the weight vector at the current iteration $\begin{pmatrix} u \\ v \end{pmatrix}^{(i)}$ we need to move the previous iterations weight vector $\begin{pmatrix} u \\ v \end{pmatrix}^{(i-1)}$ into the gradient's negative direction with the distance η . The new position of the weights $\begin{pmatrix} u \\ v \end{pmatrix}^{(i)}$ is therefore given by

$$\begin{pmatrix} u \\ v \end{pmatrix}^{(i)} = \begin{pmatrix} u \\ v \end{pmatrix}^{(i-1)} - \nabla E \cdot \eta \quad (12)$$

See `hw5-question5-7.py` for the code.

The number of iterations needed is 10, therefore the answer is d . See figure 2 for the progression of the gradient descent.

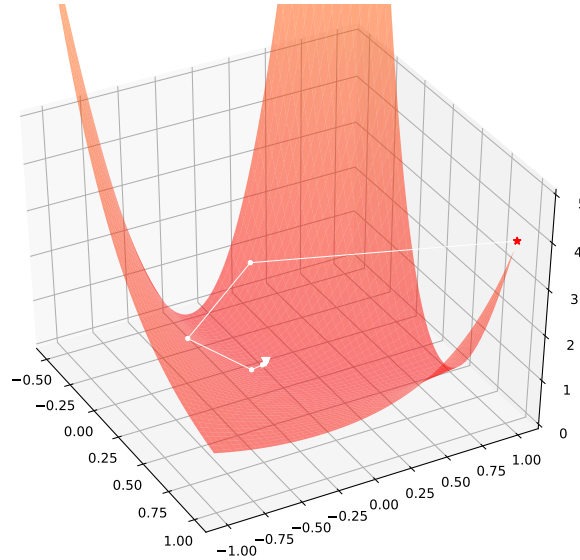


Fig. 2: Progression of the gradient descent, starting at \star and ending after 10 iterations.

QUESTION 6

What is the final value of w after reaching $E(w) < 10^{-14}$?

See `hw5-question5-7.py` for the code.

The final weight vector is $w = \begin{pmatrix} 0.045 \\ 0.024 \end{pmatrix}$. This corresponds to answer *e*.

QUESTION 7

What is the final value of $E(w)$ after 15 iterations with moving in u and v direction separately within each iteration?

Within each iteration, we do

$$\begin{pmatrix} u \\ v \end{pmatrix}^{(i)} = \begin{pmatrix} u \\ v \end{pmatrix}^{(i-1)} - \begin{pmatrix} \frac{\partial E}{\partial u} \\ 0 \end{pmatrix} \cdot \eta \quad (13)$$

followed by

$$\begin{pmatrix} u \\ v \end{pmatrix}^{(i)} = \begin{pmatrix} u \\ v \end{pmatrix}^{(i-1)} - \begin{pmatrix} 0 \\ \frac{\partial E}{\partial v} \end{pmatrix} \cdot \eta. \quad (14)$$

This is like walking around Manhattan, making only 90° turns. After 15 iterations, we have arrived at $E(w) = 2.91 \cdot 10^{-1}$. This corresponds to answer *a*. See figure 3 for the progression of the gradient descent. The final result is notably degraded in comparison to the original gradient descent from Question 5.

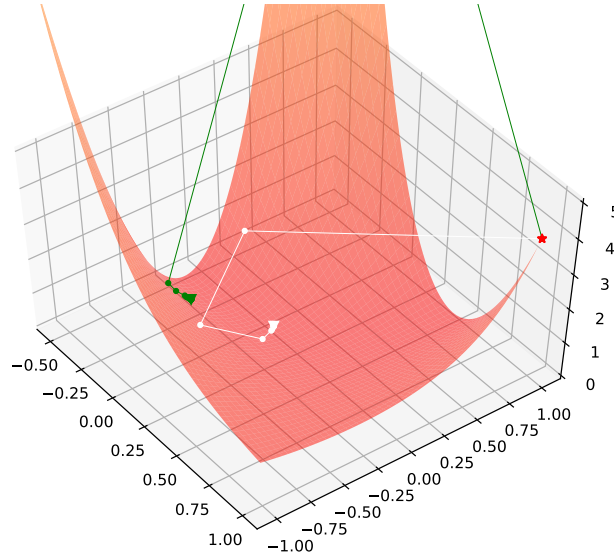


Fig. 3: Progression of the original (white line) and modified gradient descent (—), starting at \star and ending after 15 iterations.

QUESTION 8

The solution is to code the logarithmic regression with stochastic gradient descent in Python and run it for 100 runs with one training ($N = 100$) and testing ($N = 1000$) data set each. See `hw5-question8-9.py` and `learningModels.py` for the code. To answer the question, the cross entropy error of the testing sets is averaged over all data sets. The result is $E_{out} \approx 0.103$, which is closest to answer *d*.

QUESTION 9

With the same code as in Question 8, we simply average the number of iterations it takes the stochastic gradient descent to converge until $\|\mathbf{w}^t - \mathbf{w}^{t-1}\| < 0.01$ over all runs. The average number iterations was 340.52, which corresponds to answer *a*.

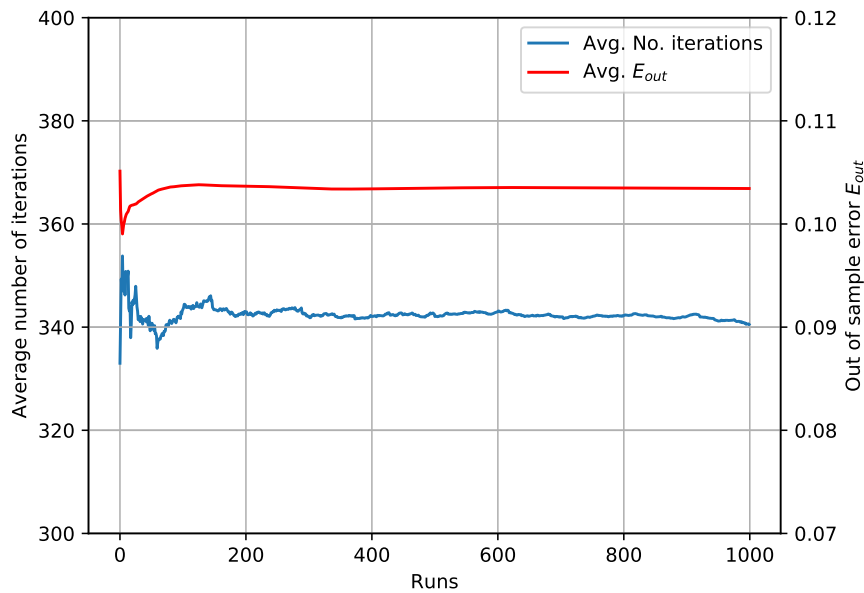


Fig. 4: Convergence of the average number of iterations that the learning algorithm took to complete the stochastic gradient descent (SGD). Left: using the same permutation of input points for each epoch of the SGD. Right: using a different permutation of input points in each epoch.

QUESTION 10

As the classifier for the perceptron is $\text{sign}(\mathbf{w}^T \cdot \mathbf{x})$, we are interested in an error function that has a zero gradient as soon as the sign of $\mathbf{w}^T \cdot \mathbf{x}$ matches that of $y = 1$ for a given data point. To see the different error measures for the case $y = 1$ in action, see figure 5. The error function of solution e features a derivative of 0 for all positive $\mathbf{w}^T \cdot \mathbf{x}$ and would therefore stop the algorithm immediately if a match between $\mathbf{w}^T \cdot \mathbf{x}$ and $y = 1$ is achieved. All other functions would not lead to immediate termination of the algorithm once the signs of $\text{sign}(\mathbf{w}^T \cdot \mathbf{x})$ and $y = 1$ are equal. The error function of choice is therefore e .

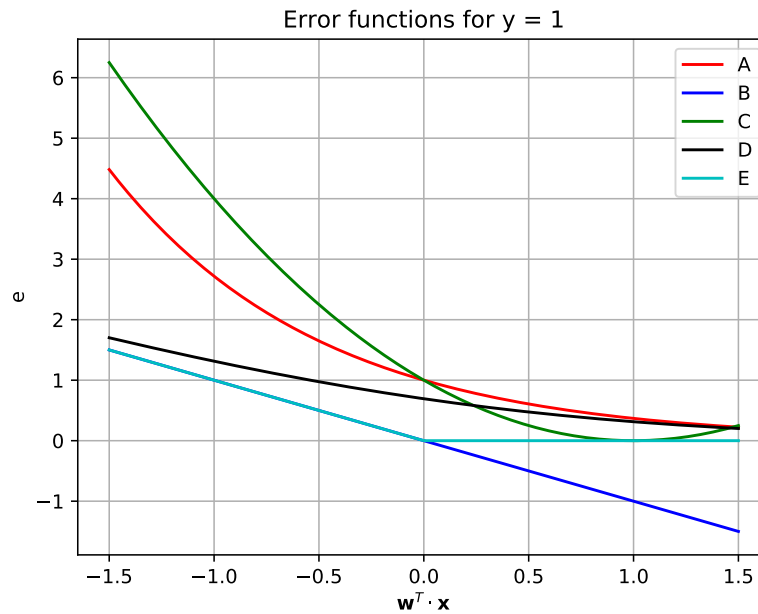


Fig. 5: Plot of the error functions for different outputs of $\mathbf{w}^T \cdot \mathbf{x}$ for $y = 1$. Error function E is the only one that would terminate the gradient descent algorithm immediately for matching signs of $\mathbf{w}^T \cdot \mathbf{x}$ and $y = 1$