Quiz 1

Question **1**

Flag question Mark 1.0 out of 1.0 Correct

Which claim is true?

Select one:

a. The test data for a machine learning model is usually assumed to be drawn from the same known distribution D that generated the training data

b. Consistent hypothesis correctly classifies all the training samples ✓

c. Version space is the space of all the hypotheses of the hypothesis class

(A) False: distribution is assumed to be unknown
(B) **True**
(C) False: version space is the space of all the **consisten** hypotheses

Question **2**

Flag question Mark 1.0 out of 1.0 Correct

Which claim is true?

Select one:

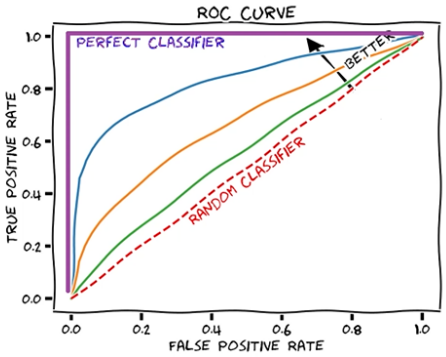
a. If a ROC curve shows a diagonal line ($x = y$), this means that the classifier performs perfectly

b. To obtain the ROC curve, the decisions of a binary classifier (classes -1 and 1) are analysed with various different thresholds θ, with decision function f as ✓

$$f(x) = \begin{cases} -1 & \text{if } P(y=1|x) < \theta \\ 1 & \text{otherwise} \end{cases}$$

c. A random classifier plotted in a ROC curve has AUC=1.

a. Incorrect. If a ROC curve shows a diagonal line (x=y), it means that the classifier performs no better than random guessing, with an AUC of 0.5.
c. Incorrect. A random classifier plotted in a ROC curve has an AUC of 0.5, which is the same as the AUC of the diagonal line.



Question **3**

Flag question Mark 1.0 out of 1.0 Correct

You are working in an engineering firm, building a machine learning system whose goal is to raise an alert for a faulty equipment based on some measurements. As the equipment is related to people's safety, is very important that no faulty equipment (=positive class) get past this model. However if some perfect products (=negative class) happen to be flagged as faulty, they can be tested further and confirmed to be in working order without too much cost.

If the precision of the system is 99.99%, how would you assess the machine learning model? (Note: recall of the next exercise is not known.)

Select one:

- a. This is a good model and performs very well for the task.
- b. You are unsure if the model is good or not: this metric does not give you enough information to make an informed assessment. ✓
- c. The model is not performing well.

b. You are unsure if the model is good or not: this metric does not give you enough information to make an informed assessment.

While precision is an important metric for assessing a binary classification model, it is not sufficient to evaluate the performance of a model in this scenario. Since the goal of the system is to catch all faulty equipment, it is important to evaluate the model's recall, or sensitivity, as well. It is possible for the model to have high precision but low recall, meaning it may not be catching all the faulty equipment. Therefore, without information on the recall of the system, it is difficult to make an informed assessment of the machine learning model.

Question **4**

Flag question Mark 1.0 out of 1.0 Correct

Continuing with the setting of the previous question, if instead you obtain recall of 99.99% (without knowing the precision), how would you assess the model then? Assume that accuracy score is also high.

Select one:

- a. This is a good model and performs very well for the task. ✓
- b. You are unsure if the model is good or not: this metric does not tell you enough information to make an informed assessment.
- c. The model is not performing well.

If the recall of a model is 99.99%, it means that the model correctly identifies almost all the positive examples (i.e., faulty equipment) in the dataset. Since the goal of the model is to catch all the faulty equipment, this is a very good indication that the model is performing well.

Question **5**

Flag question Mark 1.0 out of 1.0 Correct

Consider the diabetes dataset loaded and split into training and test sets in the provided python code (you can find the template, quiz1.py, in the Assignments page).

Fit a linear regression model to the training data without considering bias (intercept). What is the root mean squared error on the test data?

Select one:

- a. 54.58
- b. 166.14 ✓
- c. 27601.29

```
import numpy as np
from sklearn.datasets import load_diabetes
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression

# load the data
X, y = load_diabetes(return_X_y=True)

# division into training and testing
np.random.seed(42)
order = np.random.permutation(len(y))
tst = np.sort(order[:221])
tr = np.sort(order[221:])

Xtr = X[tr, :]
Xtst = X[tst, :]
Ytr = y[tr]
Ytst = y[tst]
reg = LinearRegression(fit_intercept=False).fit(Xtr, Ytr)
Ypredict = reg.predict(Xtst)
error = mean_squared_error(Ytst, Ypredict, squared=False)
print("The RMSE on the test data: ", error)
```

```
The RMSE on the test data:  166.13636616120564
```

Quiz 2

Question **1**

Flag question Mark 1.0 out of 1.0 Correct

With a PAC learnable class $\mathcal{C}$,with algorithm $\mathcal{A}$and sample S, if the generalisation error satisfies $Pr\left(R(h_S) \leq 0.1\right) \geq 0.95$, which of the statements is **true**?

Select one:

a. It is possible to run the algorithm and obtain a hypothesis with only 50% accuracy. ✓

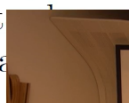
b. It is not possible to obtain a hypothesis with a worse accuracy than 90%

c. Always at least 10% of the data samples are wrongly classified.

A It is possible to run the algorithm and obtain a hypothesis with only 50% accuracy. **TRUE**: it is possible 5% of the time to obtain hypothesis with worse accuracy than 90%

B It is not possible to obtain a hypothesis with a worse accuracy than 90%. **FALSE**: This is a rare event, but possible

C Always at least 10% of the data samples are wrongly classified. **FALSE**: it's possible to get better accuracy sometimes when running the algorithm; the bound states that it is guarant
that 95% of time the generalisation error is not greater tha



Question **2**

Flag question Mark 1.0 out of 1.0 Correct

Consider classification problem on a dataset containing 4 binary features, and a binary label. Using a rule-based classifier with boolean conjunctions, the sample complexity bound is

$$m \geq \frac{1}{\epsilon}\left(\log(|\mathcal{H}|) + \log\left(\frac{1}{\delta}\right)\right).$$

You are interested in obtaining 90% accuracy in your model. Initially you are contemplating 85% confidence as you have just enough data samples for that, but would prefer 99%. How many more samples do you need to collect to obtain the better confidence?

Select one:

a. 11 more samples

b. 27 more samples ✓

c. 64 more samples

- Hypothesis class size $3^d = 3^4$: see lecture 2, slide 20

Example: Boolean conjunctions

- How many different conjunctions can be built ($=|\mathcal{H}|$)
- Each variable can appear with or without "NOT" or can be excluded from the rule = 3 possibilities
- The total number of hypotheses is thus 3^d, where d is the number of variables

- Natural logarithm
- $\epsilon = 0.1$, $\delta = 0.15$ for 85% confidence, $\delta = 0.01$ for 99%

$$\begin{aligned}
&\lceil \tfrac{1}{\epsilon}\left(\log(|\mathcal{H}|) + \log(\tfrac{1}{\delta})\right)\rceil - \lceil \tfrac{1}{\epsilon}\left(\log(|\mathcal{H}|) + \log(\tfrac{1}{\delta})\right)\rceil \\
&= \lceil \tfrac{1}{0.1}\left(\log(3^4) + \log(\tfrac{1}{0.01})\right)\rceil - \lceil \tfrac{1}{0.1}\left(\log(3^4) + \log(\tfrac{1}{0.15})\right)\rceil \\
&= 90 - 63 = 27
\end{aligned}$$

Question **3**

Flag question Mark 1.0 out of 1.0 Correct

A classifier has a VC dimension of 4. Which of the following claims is true?

Select one:

a. Given any four data points, it is possible to shatter them with the classifier.

b. It is not possible to shatter any collection of five data points with the classifier. ✓

c. It is not possible to shatter any collection of three data points with the classifier.

A Given any four data points, it is possible to shatter them with the classifier. **FALSE**: only one is enough to get this VC dimension

B It is not possible to shatter any collection of five data points with the classifier. **TRUE**

C It is not possible to shatter any collection of three data points with the classifier. **FALSE**: there is least one collection of four data points, and subsequently any of its subsets can also be shattered

Question **4**

Flag question Mark 1.0 out of 1.0 Correct

The attached example code contains a simple dataset and a binary classifier.

What is the value of the generalisation bound based on Rademacher complexity for training set sizes n=20, n=50 and n=100 (use the first $n$samples of the training set)?
Choose the closest values (randomness from calculating the empirical Rademacher bound can result in small variation in the results). Use $\delta = 0.05$.

Select one:

a. $20 \approx 0.99; 50 \approx 0.61; 100 \approx 0.42$

b. $20 \approx 1.29; 50 \approx 0.81; 100 \approx 0.65$ ✓

c. $20 \approx 1.63; 50 \approx 1.25; 100 \approx 1.12$

Question **5**

Flag question Mark 1.0 out of 1.0 Correct

Using the same setting as the previous exercise, calculate the value of the generalisation bound based on VC-dimension. The VC-dimension of perceptron is $d + 1$, where d is the number of features.

Select one:

a. $20 = 1.21; 50 = 0.85; 100 = 0.64$

b. $20 = 1.39; 50 = 0.95; 100 = 0.8$

c. $20 = 1.51; 50 = 1.05; 100 = 0.87$ ✓

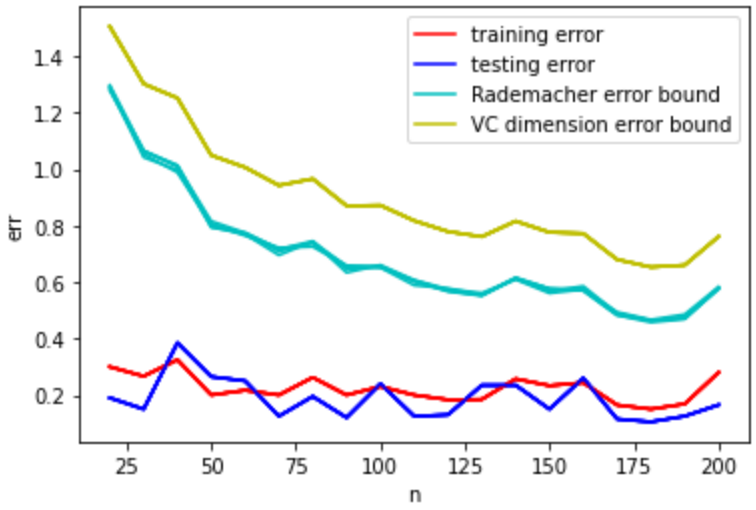
```
(using 25 random labelings to calculate R.complexity)
  n - Radem. - Vcdim
 20-  1.29 -  1.51
 50-  0.80 -  1.05
100-  0.66 -  0.87
200-  0.58 -  0.76

(using 50 random labelings to calculate R.complexity)
  n - Radem. - Vcdim
 20-  1.29 -  1.51
 50-  0.81 -  1.05
100-  0.65 -  0.87
200-  0.58 -  0.76

(using 75 random labelings to calculate R.complexity)
  n - Radem. - Vcdim
 20-  1.30 -  1.51
 50-  0.79 -  1.05
100-  0.65 -  0.87
200-  0.58 -  0.76

(using 100 random labelings to calculate R.complexity)
  n - Radem. - Vcdim
 20-  1.28 -  1.51
 50-  0.82 -  1.05
100-  0.66 -  0.87
200-  0.58 -  0.76

(using 1000 random labelings to calculate R.complexity)
  n - Radem. - Vcdim
 20-  1.29 -  1.51
 50-  0.81 -  1.05
100-  0.65 -  0.87
200-  0.58 -  0.76
```



Generalization bound based on the VC-dimension

- (Mohri, 2018) Let $\mathcal{H}$ be a family of functions taking values in $\{-1, +1\}$ with VC-dimension d. Then for any $\delta > 0$, with probability at least $1 - \delta$ the following holds for all $h \in \mathcal{H}$:

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{2\log(em/d)}{m/d}} + \sqrt{\frac{\log(1/\delta)}{2m}}$$

- $e \approx 2.71828$ is the base of the natural logarithm
- The bound reveals that the critical quantity is m/d, i.e. the number of examples divided by the VC-dimension
- Manifestation of the Occam's razor principle: to justify an increase in the complexity, we need reciprocally more data

Generalization bound with Rademacher complexity

(Mohri et al. 2018): For any $\delta > 0$, with probability at least $1 - \delta$ over a sample drawn from an unknown distribution D, for any $h \in \mathcal{H}$ we have:

$$R(h) \leq \hat{R}_S(h) + \hat{\mathcal{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}$$

The bound is composed of the sum of :

- The empirical risk of h on the training data S (with the original labels): $\hat{R}_S(h)$
- The empirical Rademacher complexity: $\hat{\mathcal{R}}_S(\mathcal{H})$
- A term that tends to zero as a function of size of the training data as $O(1/\sqrt{m})$ assuming constant δ.

```python
def get_rademacher_bound(eerror, erc, n, delta):
    """
    param eerror: empirical error
    param erc: empirical rademacher complexity
    param n: numer of training samples
    param delta: delta
    """
    return eerror + erc + 3 * np.sqrt(np.log(2/delta)/(2*n))

def get_vc_bound(eerror, vcd, n, delta):
    """
    param eerror: empirical error
    param vcd: the vc dimension
    param n: numer of training samples
    param delta: delta
    """
    from math import e
    return eerror + np.sqrt((2 * vcd * np.log((e * n)/vcd))/n) +
np.sqrt((np.log(1/delta))/(2 * n))
```

Quiz 3

Question **1**

Flag question Mark 1.0 out of 1.0 Correct

Let x be chosen from the interval $[-1,+1]$, and the labels, y, from the set $\{0,1\}$. We are given the following conditional probabilities for all x and y:

$$Pr(1|x) = \begin{cases} +x+1 & x \in [-1,0] \\ -x+1 & x \in [0,+1] \end{cases}$$

$$Pr(0|x) = \begin{cases} -x & x \in [-1,0] \\ +x & x \in [0,+1] \end{cases}$$

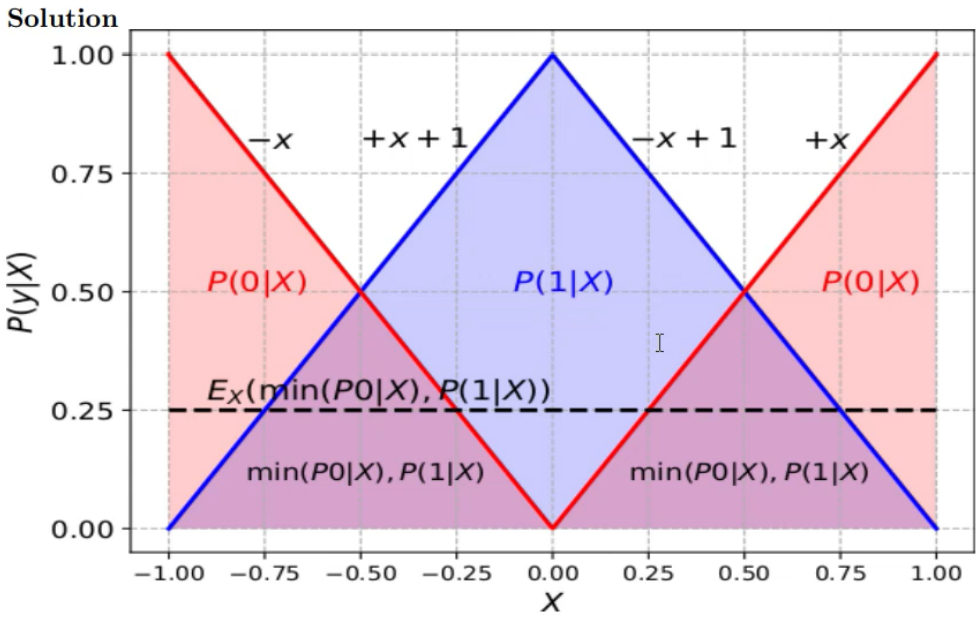
The question: what is the Bayes error of the Bayes classifier relating to this model if x is uniformly distributed on $[-1,+1]$?

Hint: You might solve the problem by drawing a plot representing the functions of the conditional probabilities.

Select one:

- a. 0.75
- b. 0.5
- c. 0.3
- d. 0.25 ✓

Solution



$$\text{noise}(x) = \min(P(0|x), P(1|x)) = \begin{cases} +x+1 & x \in [-1, -0.5] \\ -x+0 & x \in (-0.5, 0] \\ +x+0 & x \in [0, 0.5] \\ -x+1 & x \in (0.5, 1] \end{cases}$$

If the distribution of x is uniform, then the density of x, $f(x) = 0.5$, therefore we have

$$\begin{aligned} &\mathrm{E}_x(\text{noise}(x)) = \textstyle\int_{-1}^{+1} \text{noise}(x) f(x) dx \\ &= \int_{-1}^{-0.5} 0.5(x+1)\, dx + \int_{-0.5}^{0} 0.5(-x)\, dx + \int_{0}^{0.5} 0.5x\, dx + \int_{0.5}^{1} 0.5(-x+1)\, dx = \boxed{0.25}. \end{aligned}$$

Bayes error and noise

- A hypothesis with $R(h) = R^*$ is called the **Bayes classifier**
- The Bayes classifier can be defined in terms of conditional probabilities as

$$h_{Bayes}(x) = \text{argmax}_{y \in \{0,1\}} Pr(y|x)$$

- The average error made by the Bayes classifer at $x \in X$ is called the **noise**

$$noise(x) = \min(Pr(1|x), Pr(0|x))$$

- Its expectation $E(noise(x)) = R^*$ is the Bayes error
- Similarly to the Bayer error, Bayes classifier is a theoretical tool, not something we can compute in practice

Question **2**

Flag question Mark 1.0 out of 1.0 Correct

In applying the Perceptron algorithm on a data set, $\{(\mathbf{X}_i, y_i)\}_{i=1}^m)$, $y_i = -1, +1, \forall i$, we might scale all input vectors $\{\mathbf{x}_i\}_{i=1}^m$ with a positive constant $\lambda > 0$. This issue arises when the input vectors are normalized in a learning problem. What is the effect of this scaling? Assume that the two classes are linearly separable, and no bias term is included into the predictor function. Select that answer which is true if this kind of scaling is applied!

Select one:

a. The weigh vector $\mathbf{w}$ is the same when the algorithm is stopped.

b. In the Novikoff's Theorem, the number of expected iterations t does not depend on the scaling. It is assumed that the margin γ is the largest value satisfying $y_i \mathbf{w}_* \mathbf{x}_i \geq \gamma$ for all $i = 1, \ldots, m$. ✓

c. In the Novikoff's Theorem, the number of expected iterations t increases when the λ increases.

d. In the Novikoff's Theorem, the number of expected iterations t decreases when the λ increases.

Solution

Input: Training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m, \mathbf{x} \in \mathbb{R}^d, y \in \{-1, +1\}$
Initialize $w^{(1)} \leftarrow (0, \ldots, 0), t \leftarrow 1$, *stop* $\leftarrow$ *FALSE*
repeat
  if exists i, s.t. $y_i {\mathbf{w}^{(t)}}^T \mathbf{x}_i \leq 0$ **then**
    $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + y_i \mathbf{x}_i$
  else
    stop $\leftarrow$ *TRUE*
  end if
  $t \leftarrow t + 1$
until *stop*

- The perceptron algorithm can be shown to eventually converge to a consistent hyperplane if the two classes are **linearly separable**, that is, if there exists a hyperplane that separates the two classes
- Theorem (Novikoff):
  - Let $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ be a linearly separable training set.
  - Let $R = \max_{\mathbf{x}_i \in S} \|\mathbf{x}_i\|$.
  - Let there exist a vector $\mathbf{w}_*$ that satisfies $\|\mathbf{w}_*\| = 1$ and $y_i \mathbf{w}_*^T \mathbf{x}_i + b_{opt} \geq \gamma$ for $i = 1 \ldots, m$.
  - Then the perceptron algorithm will stop after at most $t \leq (\frac{2R}{\gamma})^2$ iterations and output a weight vector $\mathbf{w}^{(t)}$ for which $y_i \mathbf{w}^{(t)} \mathbf{x}_i \geq 0$ for all $i = 1 \ldots, m$

- The inequality $y_i {\mathbf{w}^{(t)}}^T \lambda \mathbf{x}_i \leq 0$ remains TRUE for any $\lambda > 0$.
- $\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)} + y_i \lambda \mathbf{x}_i$ increases the length of $\mathbf{w}$ by λ but the direction is preserved.

- R is scaled by λ, λR.
- $||\mathbf{w}_*|| = 1$ is unchanged, since the direction is preserved.
- $y_i \mathbf{w}_* \lambda \mathbf{x}_i$ is scaled by λ, thus γ is scaled as well, $\lambda\gamma$, since γ is the maximum computed on all i.
- Finally: $\left(\frac{\lambda 2R}{\lambda \gamma}\right)^2 =$

Question **3**

Flag question Mark 2.0 out of 2.0 Correct

Let the stochastic gradient algorithm for Logistic Regression be applied to find the best classifier on the Breast Cancer dataset of the Sklearn package. That algorithm is presented in Lecture 5. In the learning 5-fold cross validation needs to be used on the data. To select the folds, the KFold method of the Sklearn should be used with the parameters shown in the program example below. The labels of the Breast Cancer dataset are of $\{0, 1\}$ which need to be converted into $\{-1, +1\}$. Normalize the rows of the input matrix to have the L_2 norm to be equal to 1. *Without that normalization overflow error can occur in the exponential function!*

The training examples are processed in the order of the original data file, within each fold. The learning parameters, number of iteration and the learning speed, need to be chosen as shown in the program example. The implementation might start with these lines:

```
----------------------
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import KFold

# load the data
X, y = load_breast_cancer(return_X_y=True)  ## X input, y output
## to convert the {0,1} output into {-1,+1}
y = 2*y - 1

## learning parameters
nitermax = 50  ## maximum iteration
eta = 0.1      ## learning speed

nfold = 5         ## number of folds

## to split the data into 5-folds we need
cselection = KFold(n_splits=nfold, random_state=None, shuffle=False)
----------------------
```

- The gradient is the vector of partial derivatives of the objective function $J(\mathbf{w})$ with respect to all parameters w_j

$$\nabla J(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \nabla J_i(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \left[\frac{\partial}{\partial w_1} J_i(\mathbf{w}), \ldots, \frac{\partial}{\partial w_d} J_i(\mathbf{w})\right]^T$$

- Compute the gradient by using the regular rules for differentiation. For the logistic loss we have

$$\begin{aligned}\frac{\partial}{\partial w_j} J_i(\mathbf{w}) &= \frac{\partial}{\partial w_j} \log(1 + \exp(-y_i \mathbf{w}^T x_i)) = \frac{\exp(-y_i \mathbf{w}^T x_i)}{1 + \exp(-y_i \mathbf{w}^T x_i)} \cdot (-y_i x_{ij}) \\ &= -\frac{1}{1 + \exp(y_i \mathbf{w}^T x_i)} y_i x_{ij} = -\phi_{logistic}(-y_i \mathbf{w}^T x_i) y_i x_{ij}\end{aligned}$$

Initialize $\mathbf{w} = 0$

repeat

Draw a training example (x_i, y_i) uniformly at random

Compute the update direction corresponding to the training example: $\Delta\mathbf{w} = -\nabla J_i(\mathbf{w})$

Determine a stepsize η

Update $\mathbf{w} = \mathbf{w} - \eta \nabla J_i(\mathbf{w})$

until stopping criterion statisfied

Output $\mathbf{w}$

The task is to run the Logistic Regression algorithm on the 5-fold cross-validation, and compute the F1-score on the corresponding test sets. In selecting the training and the test sets you might follow the example code of the KFold function provided in the Sklearn.

The question: what is the average F1 score computed on the 5-folds? Round the numbers up to 2 decimals, and take the closest one. *Be aware, the Logistic Regression algorithm of Lecture 5 is not the same which is implemented in the Sklearn. use that version which is presented in the Lecture!*

Select one:

- a. 0.89 ✓
- b. 0.95
- c. 0.99
- d. 0.70

Question **4**

Flag question Mark 1.0 out of 1.0 Correct

In the previous question (Question 3), where the stochastic gradient algorithm is applied for the Logistic Regression, scale each of the input variables, columns, to have the maximum absolute value equal to 1. Repeat the same learning procedure of Question 3. In each fold also compute the maximum functional margin that the training can achieve.

The question: what is the average F1-score after scaling the input variables, columns, and what is the average functional margin? The averages are computed on all folds. Similarly to Question 3 round the numbers and find the closest case.

Select one:

- a. 0.70, 29.30
- b. 0.99, 45.10
- c. 0.85, 20.90
- d. 0.95, 23.20 ✓

Quiz 4

Question **1**

Flag question Mark 1.0 out of 1.0 Correct

In Lecture 7, Kernel methods, it is discussed how to build new kernels from known ones, see Slide *Several ways to get to a kernel*. Assume that we have a kernel function $\kappa(.,.)$ which satisfies the following condition $\kappa(\mathbf{x},\mathbf{z}) \leq \rho < 1$ for any pair $\mathbf{x}$ and $\mathbf{z}$. Let a new function be $\kappa^*(\mathbf{x},\mathbf{z}) = 1/(1-\kappa(\mathbf{x},\mathbf{z}))$. What could we say about the function κ^*?

Hint: Think about which of the kernel constructing operations might relate to this question. Could the new kernel be transformed into a sequence?

Select one:

- a. κ^* can not be expressed by the kernel constructing operations.
- b. κ^* is not positive(semi) definite.
- c. κ^* is a valid positive(semi) definite kernel. ✓
- d. κ^* is a valid kernel positive(semi) definite kernel if $\rho > 1$ only.

Solution:

The corresponding slide:

Several ways to get to a kernel

Approach IV. Making kernels from kernels

- Examples of elementary operations that give valid kernels when applied to kernels κ_n, $n = 1, 2, \ldots$
  1. Sum: $\kappa(\mathbf{x},\mathbf{z}) = \kappa_1(\mathbf{x},\mathbf{z}) + \kappa_2(\mathbf{x},\mathbf{z})$
  2. Scaling with a positive scalar: $\kappa(\mathbf{x},\mathbf{z}) = a\kappa_1(\mathbf{x},\mathbf{z}), a > 0$
  3. Itemwise product: $\kappa(\mathbf{x},\mathbf{z}) = \kappa_1(\mathbf{x},\mathbf{z})\kappa_2(\mathbf{x},\mathbf{z})$
  4. Normalization: $\kappa(\mathbf{x},\mathbf{z}) = \frac{\kappa_1(\mathbf{x},\mathbf{z})}{\sqrt{\kappa_1(\mathbf{x},\mathbf{x})\kappa_1(\mathbf{z},\mathbf{z})}} = \left\langle \frac{\phi(\mathbf{x})}{\|\phi(\mathbf{x})\|}, \frac{\phi(\mathbf{z})}{\|\phi(\mathbf{z})\|} \right\rangle$
  5. Pointwise limit: $\kappa(\mathbf{x},\mathbf{z}) = \lim_{n\to\infty} \kappa_n(\mathbf{x},\mathbf{z})$
  6. Composition with a power series of radius of convergence ρ: $\kappa(\mathbf{x},\mathbf{z}) = \sum_{n=0}^{\infty} a_n \kappa(\mathbf{x},\mathbf{z})^n$, with $a_n \geq 0$ for all n, and $|\kappa(\mathbf{x},\mathbf{z})| < \rho$
- The operations can be combined to construct arbitrarily complex kernels, e.g. polynomial kernels and Gaussian kernels can be derived this way (see details in the Mohri book ch. 6)

If $|\kappa(\mathbf{x},\mathbf{z})| \leq \rho < 1$ then $\kappa^*(\mathbf{x},\mathbf{z}) - 1/(1-\kappa(\mathbf{x},\mathbf{z}))$ can be written as a geometric series, namely $\boxed{\kappa^*(\mathbf{x},\mathbf{z}) = 1/(1-\kappa(\mathbf{x},\mathbf{z})) = \sum_{n=0}^{\infty} \kappa(\mathbf{x},\mathbf{z}))^n}$.

Now we can apply Rule 6 from Slide "Several ways to get to a kernel" of Lecture 7 by setting $a_n = 1$ for all n and with the convergence radius $\rho < 1$.

Question **2**

Flag question Mark 1.0 out of 1.0 Correct

A Gaussian(RBF) kernel has this form $\kappa_{rbf}(\mathbf{x},\mathbf{z}) = \exp(-||\mathbf{x}-\mathbf{z}||^2/(2\sigma^2))$. Let $\kappa(\mathbf{x},\mathbf{z}) = \langle\phi(\mathbf{x}),\phi(\mathbf{z})\rangle$ be any valid kernel with feature map ϕ.

Let us try to construct a new kernel out of the Gaussian one, namely we have $\kappa_{rbf\_mod}(\mathbf{x},\mathbf{z}) = \exp(-(\kappa(\mathbf{x},\mathbf{x}) + \kappa(\mathbf{z},\mathbf{z}) - 2\kappa(\mathbf{x},\mathbf{z}))/(2\sigma^2))$. Which of these statements is true?

Hint: How is the Gaussian kernel function defined? How is it built on the underlying Hilbert space?

Select one:

a. κ_{rbf_mod} is only a valid kernel if κ is a Gaussian one.

b. κ_{rbf_mod} is a valid, positive semi-definite kernel for any κ. ✓

c. There are cases where κ_{rbf_mod} is not positive semi-definite.

Solution:

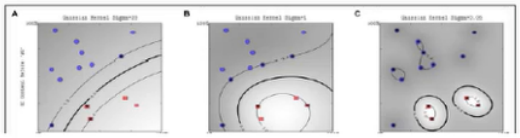
The corresponding slides:

Non-linear kernels: Gaussian kernel (Radial basis function kernel, RBF)

Gaussian kernel between two inputs $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$ with bandwidth parameter $\sigma > 0$:

$$\kappa_{RBF}(\mathbf{x},\mathbf{z}) = \exp\left(-\|\mathbf{x}-\mathbf{z}\|^2/(2\,\sigma^2)\right)$$

- Large values for σ give a smoother kernel, slower decay of kernel values as a function of the squred euclidean distance $\|\mathbf{x}-\mathbf{z}\|^2$, and a more linear decision boundary
- Small values for σ give a less smooth kernel, faster decay of kernel values as a function of the squared euclidean distance $\|\mathbf{x}-\mathbf{z}\|^2$, and a more non-linear decision boundary



24

A Gaussian(RBF) kernel is defined by $\kappa_{rbf}(\mathbf{x},\mathbf{z}) = \exp(-||\mathbf{x}-\mathbf{z}||^2/(2\sigma^2))$, where $\mathbf{x}$ and $\mathbf{z}$ are taken from an arbitrary Hilbert space $\mathcal{H}$. The expression, $||\mathbf{x}-\mathbf{z}||^2$, is the square distance between $\mathbf{x}$ and $\mathbf{z}$ in $\mathcal{H}$.

If the kernel κ is defined by the feature map ϕ, in the corresponding Hilbert space $\mathcal{H}_\phi$ the square distance between $\phi(\mathbf{x})$ and $\phi(\mathbf{z})$ is given by $||\phi(\mathbf{x})-\phi(\mathbf{z})||^2$. It can give an RBF kernel on $\mathcal{H}_\phi$ by

$\boxed{\kappa_{rbf_\phi}(\mathbf{x},\mathbf{z}) = \exp(-||\phi(\mathbf{x})-\phi(\mathbf{z})||^2/(2\sigma^2))}$.

Let us unfold the square distance $||\phi(\mathbf{x})-\phi(\mathbf{z})||^2$.

$$\begin{aligned}\boxed{||\phi(\mathbf{x})-\phi(\mathbf{z})||^2} &= \langle\phi(\mathbf{x})-\phi(\mathbf{z}, \phi(\mathbf{x})-\phi(\mathbf{z})\rangle \\ &= \langle\phi(\mathbf{x}),\phi(\mathbf{x})\rangle + \langle\phi(\mathbf{z}),\phi(\mathbf{z})\rangle - 2\langle\phi(\mathbf{x}),\phi(\mathbf{z})\rangle \\ &= \boxed{\kappa(\mathbf{x},\mathbf{x}) + \kappa(\mathbf{z},\mathbf{z}) - 2 * \kappa(\mathbf{x},\mathbf{z})}.\end{aligned}$$

It means that

$\boxed{\kappa_{rbf_\phi}(\mathbf{x},\mathbf{z}) = \exp(-||\phi(\mathbf{x})-\phi(\mathbf{z})||^2/(2\sigma^2)) = \exp(-(\kappa(\mathbf{x},\mathbf{x}) + \kappa(\mathbf{z},\mathbf{z}) - 2\kappa(\mathbf{x},\mathbf{z}))/(2\sigma^2)) = \kappa_{rbf\_mod}(\mathbf{x},\mathbf{z})}$,

therefore $\kappa_{rbf\_mod}(\mathbf{x},\mathbf{z})$ is valid, positive semidefinite kernel.

Question **3**

Flag question Mark 1.0 out of 1.0 Correct

Let a polynomial kernel, $\kappa_{pol}(\mathbf{x}, \mathbf{z})$ be represented explicitly $(\mathbf{x}^T\mathbf{z})^q$ where q is the degree. What is the dimension of the explicit feature vector? Assume that the degree of the polynomial is 3, and the polynomial kernel is defined on the vector space of dimension 3.

Be careful, there is no constant term in this kernel definition! How can the feature dimension change if the constant is not included?

Select one:

- a. 15
- b. 20
- c. 10 ✓
- d. 30

The corresponding slides:

Non-linear kernels: Polynomial kernel

- Given inputs $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$, the **polynomial kernel** is given by

$$\kappa_{pol}(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z} + c)^q$$

- Integer $q > 0$ gives the **degree** of the polynomial kernel
- Real value $c \geq 0$ is a weighting factor for lower order polynomial terms
- The underlying features are **non-linear**: monomial combinations $x_1 \cdot x_2 \cdots x_k$ of degree $k \leq q$ of the original features x_j

Example: Polynomial kernel on 2D inputs

- Consider two-dimensional inputs $\mathbf{x} = [x_1, x_2]^T \in \mathbb{R}^2$
- The second degree polynomial kernel is given by $\kappa(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^2$
- We can write it as a inner product in $\mathbb{R}^6$:

$$\begin{aligned}
\kappa(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^2 &= (x_1x_1' + x_2x_2' + c)^2 = \\
&= x_1x_1'x_1x_1' + x_2x_2'x_2x_2' + c^2 + \\
&+ 2x_1x_1'x_2x_2' + 2cx_1x_1' + 2cx_2x_2' \\
&= \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2c}x_1 \\ \sqrt{2c}x_2 \\ c \end{bmatrix}^T \begin{bmatrix} x_1'^2 \\ x_2'^2 \\ \sqrt{2}x_1'x_2' \\ \sqrt{2c}x_1' \\ \sqrt{2c}x_2' \\ c \end{bmatrix} \\
&= \phi(\mathbf{x})^T\phi(\mathbf{x}'),
\end{aligned}$$

where $\phi(\mathbf{x}) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2c}x_1, \sqrt{2c}x_2, c]^T$

Non-linear kernels: Polynomial kernel

- The dimension of the polynomial feature space is $\binom{d+q}{q} = O((d+q)^q)$ where d is the dimension of the input space X and q is the degree of the polynomial.
- Explicitly maintaining the feature map $\phi(\mathbf{x})$ and the weight vector $\mathbf{w}$, and evaluating the model $\mathbf{w}^T\phi(\mathbf{x})$ takes $O(d^q)$ time and space
- However, the polynomial kernel $\kappa(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^q$ can be computed in time $O(d)$ in preprocessing, and evaluated in constant time
- Evaluating the model using the dual representation $\sum_{i=1}^{m} \alpha_i y_i \kappa(\mathbf{x}_i, \mathbf{x})$ takes $O(m)$ time
- Trade-off: No computational overhead from working in the high-dimensional feature space, but linear dependency in the size of training data

Let us start at Slide "Example: Polynomial kernel on 2D inputs" in Lecture 7. On that slide, the degree 2 case is shown. From that case we can go to the more general situation. First the case when the constant is included. We have that $(\mathbf{x}^T\mathbf{z} + c) = ([\sqrt{c}, \mathbf{x}], [\sqrt{c}, \mathbf{z}])$. For dimension 2, the component wise form is $([\sqrt{c}, x_1, x_2], [\sqrt{c}, z_1, z_2])$. Let $\sqrt{(c)}$ be indexed by 0, and we have $([x_0, x_1, x_2], [z_0, z_1, z_2])$, where $x_0 = z_0 = \sqrt{c}$. Now we can write by the help of the known formula of the multinomial coefficients

$$([x_0, x_1, x_2], [z_0, z_1, z_2])^4 = \left(\sum_{i=0}^{2} x_i z_i\right)^4 = \sum_{k_1+k_2+k_3+k_4=4} \binom{4!}{k_1!k_2!k_3!k_4!} \prod_{i=0}^{2} (x_i z_i)^{k_i}, \tag{1}$$

where $k_1, k_2, k_3, k_4 \in \{0, 1, 2, 3, 4\}$.

The description of this formula can be found, for example https://dlmf.nist.gov/26.4 on the web site of the National Institute of Standards and Technology of the US Government,

or https://en.wikipedia.org/wiki/Multinomial_theorem, and several other combinatorial reference sites.

Starting at (1) we can decompose the product

$$\sum_{k_1+k_2+k_3+k_4=4} \binom{4!}{k_1!k_2!k_3!k_4!} \prod_{i=0}^{2} (x_i z_i)^{k_i} = \sum_{k_1+k_2+k_3+k_4=4} \binom{4!}{k_1!k_2!k_3!k_4!} \prod_{i=0}^{2} (x_i)^{k_i} \prod_{i=0}^{2} (z_i)^{k_i}$$

$$= \left(\sum_{k_1+k_2+k_3+k_4=4} \underbrace{\left(\sqrt{\binom{4!}{k_1!k_2!k_3!k_4!}} \prod_{i=0}^{2} (x_i)^{k_i}\right)}_{\phi(\mathbf{x})_{k_1,k_2,k_3,k_4}} \underbrace{\left(\sqrt{\binom{4!}{k_1!k_2!k_3!k_4!}} \prod_{i=0}^{2} (z_i)^{k_i}\right)}_{\phi(\mathbf{z})_{k_1,k_2,k_3,k_4}} \right) = (\phi(\mathbf{x}), \phi(\mathbf{z})).$$

Hence, the feature vector $\phi(\mathbf{x})$ has as many components as the number of different sums of $k_1, k_2, k_3 \in \{0, 1, 2\}$ when they satisfy the constraint $k_1 + k_2 + k_3 - 3$, That number is $\binom{3+3}{3}$ which gives 20.

Following the same line, the general case when the dimension of the vector space is N, the number of components in the polynomial feature vector for degree d is equal to $\boxed{\binom{N+d}{d}}$, see Slide 23.

You can also find a similar description in the lecture book "Understanding Machine Learning: From Theory to Algorithms", Section 16.2, Page 220, and in Example 6.4 on Page 108 in the other lecture book "Foundations of Machine Learning ".

Now we can deal with the case when the constant is dropped. If there is no constant then all terms with degree less than d will not occur in unfolded expression of the polynomial kernel.

- All terms have degree d, which means the polynomial is homogeneous.
- The number of terms of degree less than d with N variables is equal to $\boxed{\binom{N+d-1}{d-1}}$.
- Thus the case when the constant is left out contains $\boxed{\binom{N+d}{d}} - \boxed{\binom{N+d-1}{d-1}} = \boxed{\binom{N+d-1}{d}}$ terms.
- In the case where $N = 3$ and $d = 3$ we have $\boxed{\binom{3+3-1}{3} = 10}$.

Question **4**

Flag question Mark 2.0 out of 2.0 Correct

In this question the behavior of algorithms developed to solve the Support Vector Machine problem are compared. The comparison is carried out on the breast cancer dataset of the sklearn package,

```
from sklearn.datasets import load_breast_cancer
```

The labels of the Breast Cancer dataset are of $\{0, 1\}$ which need to be converted into $\{-1, +1\}$. Scale each of the input variables to have the maximum absolute value equal to 1. For example, these steps can be implemented by

```
import numpy as np

from sklearn.datasets import load_breast_cancer

from sklearn.svm import SVC

from sklearn.metrics import roc_auc_score


# load the data

X, y = load_breast_cancer(return_X_y=True)  ## X input, y output

print(X.shape, y.shape)
```

```
# load the data

X, y = load_breast_cancer(return_X_y=True)  ## X input, y output

print(X.shape, y.shape)


## to convert the {0,1} output into {-1,+1}

y = 2 * y -1


## X is the input matrix

mdata,ndim = X.shape

## normalization by L infinity norm

X/= np.outer(np.ones(mdata),np.max(np.abs(X),0))


## number of iteration

niter = 10


## penalty constant for the of the Stochastic Dual Coordinate Ascent algorithm

C = 1000
```

Three methods need to be applied. Those methods are the following ones:

- The *Stochastic Dual Coordinate Ascent for SVM* algorithm which is described in the lecture slides. In this algorithm, use the linear kernel on the top of the input data. The penalty constant, C, is equal to 1000. Repeat 10 times the algorithm on the full training dataset.
- The method *sklearn.svm.SVC* from the sklearn package with polynomial kernel, denoted by *poly* in the sklearn. The degree of the polynomial is 5, and the penalty constant, C, here is equal to 1, the default.
- The method *sklearn.svm.SVC* again from the sklearn package with Gaussian kernel, denoted by *rbf* in the sklearn. The *gamma* parameter of the rbf kernel is set to *scale*, and the penalty constant, C, here is also equal to 1, the default.

Process the data in the original order of the examples appearing in the data set. No training and test split is applied. The original full data is used in the training and in the test phases as well.

Compute the Area Under the Curve(AUC) score for the three versions of the SVM solution methods, thus for the *Stochastic Dual Coordinate Ascent for SVM*, the *sklearn.svm.SVC(kernel = "poly")* and the *sklearn.svm.SVC(kernel = "rbf")*. Let the short names of the methods be SDCA, SVCPoly, SVCRbf.

Question: What is the order of the performances of the methods measured by Area Under the Curve(AUC)?

Hint: You might apply the *sklearn.metrics.roc_auc_score* function to compute the AUC score. The sklearn also contains other alternative approaches.

Select one:

- a. $SDCA < SVCPoly < SVCRbf$
- b. $SDCA < SVCRbf < SVCPoly$ ✓
- c. $SVCRbf < SVCPoly < SDCA$
- d. $SVCRbf < SDCA < SVCPoly$

The corresponding slides:

Dual Soft-Margin SVM

A dual optimization problem for the soft-margin SVM with kernels is given by

$$\begin{array}{rc} \textit{Maximize} & OBJ(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \\ \textit{w.r.t} & \text{variables } \boldsymbol{\alpha} \in \mathbb{R}^m \\ \text{Subject to} & 0 \leq \alpha_i \leq C/m \\ & \text{for all } i = 1, \ldots, m \end{array}$$

- It is a QP with variables α_i, again with a unique optimum
- At optimum, will have implicitly computed the optimal hyperplane $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$
- The data only appears through the kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Full derivation requires techniques of optimization theory, which we will skip here

Kernel trick

- We can consider transformations of the input with some basis functions $\phi : \mathbb{R}^d \mapsto \mathbb{R}^k$
- The optimal hyperplane $\mathbf{w} \in \mathbb{R}^k$ will satisfy: $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \phi(\mathbf{x}_i)$
- Assume κ_ϕ computes an inner product in the space $\kappa_\phi(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$
- Then can compute the hyperplane in the transformed space

$$\mathbf{w}^T \phi(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i \kappa_\phi(\mathbf{x}_i, \mathbf{x})$$

  and the squared norm of the weight vector $\|\mathbf{w}\|^2 = \mathbf{w}^T \mathbf{w} = \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa_\phi(\mathbf{x}_i, \mathbf{x}_j)$
- We do not need to explicitly refer to the transformed data $\phi(\mathbf{x})$ or the weight vector $\mathbf{w}$, both of which could be high-dimensional
- This is sometimes called the **kernel trick**

Stochastic Dual Coordinate Ascent for SVM

Initialize $\boldsymbol{\alpha} = \mathbf{0}$
repeat
  Select a random training example (x_i, y_i)
  Update the dual variable: $\alpha_i = \frac{1 - y_i \sum_{j \neq i} \alpha_j y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)}{\kappa(\mathbf{x}_i, \mathbf{x}_i)}$
  Clip to satisfy the constraints: $\alpha_i = \min(C/m, \max(0, \alpha_i))$
until stopping criterion is satisfied
return $\boldsymbol{\alpha}$

Quiz 5

Question **1**

Flag question Mark 1.0 out of 1.0 Correct

Which option is NOT a valid statement relating to the Multi-layer perceptrons.

Select one:

a. If the activation functions are linear in all layers then that network can be replaced with one containing only a linear function.

b. The function NOT(XOR(.,.)) can not be represented by perceptron of one layer.

c. It is not NP-hard to find the parameters minimizing the empirical error for a network with one single hidden layer that contains 5 neurons. ✓

d. Binary classification can be realized by applying softmax-functio

1 - Yes

Why do we need non-linear activation functions?

- Consider having two layer network with first layer computing $z_h = \sum_j w_{hj} x_j$ and the second layer computing $y_i = \sum_j v_{ih} z_h$
- The total function is thus:

$$y_i = \sum_h v_{ih} \sum_j w_{hj} x_j = \sum_j \sum_h v_{ih} w_{hj} x_j$$

- We can compute the same with a linear function:

$$y_i = \sum_j u_{ij} x_j$$

where $u_{ij} = \sum_h v_{ih} w_{hj}$

- Thus there is no real non-linearity in the model and our model reduces to learning a linear hyperplane

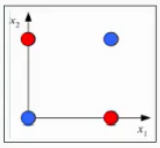
To make the network structure useful, we need non-linear activation functions

2 - Yes

XOR with perceptron

- The exclusive or, or XOR operator cannot be represented by the perceptron
- This is because the output XOR function is not linearly separable: there is no hyperplane that can separate $(0,0),(1,1)$ from $(0,1),(1,0)$

x_1	x_2	r
0	0	0
0	1	1
1	0	1
1	1	0



3 - Not

The bad news: hardness of training MLPs

Learning optimal weights for MLPs and other neural networks is computationally hard (Shalev-Shwartz and Ben-David, 2014):

- It is NP-hard to find the parameters that minimizes the empirical error, for a network with a single hidden layer that contains 4 neurons or more
- Even close-to-minimal error is NP-hard to achieve
- Changing the structure of the network is not likely to make learning easier, since any function class that can represent intersections of halfspaces is NP-hard under some cryptographic assumptions

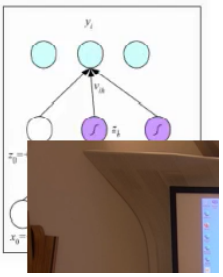
Thus in practice we need to resort in heuristic optimization approaches with no theoretical guarantees of optimality

4 - Yes

Backpropagation algorithm for classification tasks

The backpropagation algorithm described can be adapted for classification tasks:

- For binary classification task we change the output activation function to sigmoid function, either logistic (with 0/1 labels) or tanh ($-1/+1$ labels)
- Multiclass classification can be implemented by using K output units and applying a softmax-function $y_i = \frac{\exp(w_i^T z)}{\sum_k \exp(w_k^T z)}$



Question **2**

Flag question Mark 1.0 out of 1.0 Correct

Assume 25 base learners with independent true risk 0.45 ($\epsilon = 0.45$), are grouped to form a strong learner via majority voting. What is the probability of having an incorrect aggregated prediction (using the formula given on the slide "Why can model combination work? A thought experiment" of Lecture about "Ensemble Learning"):

Select one:

- a. 0.50
- b. 0.31 ✓
- c. 0.15
- d. 0.22

Solution:

Slide 4 of Lecture 9:

Why can model combination work? A thought experiment

Table shows the behaviour of the true risk of the ensemble with different values of L and ϵ

$$R(h) = P(h(\mathbf{x}) \neq C(\mathbf{x})) = \sum_{k=\lceil L/2 \rceil}^{L} \binom{L}{k} \epsilon^k (1-\epsilon)^{L-k}$$

- With low values of ϵ the risk goes down quickly to zero
- Even with $\epsilon = 0.45$ the risk is low with large enough L
- With $\epsilon = 0.5$ (i.e. random guessing) the risk remains at 0.5 independent of L

L	ϵ =0.1	0.33	0.45	0.5
5	0.0086	0.2050	0.4069	0.5000
11	0.0003	0.1171	0.3669	0.5000
101	0.0000	0.0002	0.1562	0.5000
501	0.0000	0.0000	0.0124	0.5000

Code example:

```
import numpy as np
from scipy.stats import binom

L =25        ## number of learners
eps = 0.45   ## true risk

Rh = 0
for k in range(int(np.ceil(L/2)),L+1):
  Rh += binom.pmf(k,L,eps)

print(Rh)
```

Question **3**

Flag question Mark 1.0 out of 1.0 Correct

Which option is NOT a valid statement relating to the ensemble learning.

Select one:

a. The loss function of the LogitBoost is not equal to logarithm of the loss of the AdaBoost

b. The sum of the weights in the AdaBoost satisfies these rules: $\sum_{i=1}^{m} D_t(i) = 1$ and $D_t > 0$ for any t. Thus they can be interpreted as a probability distribution on the training examples.

c. An ensemble of learners taken from a hypothesis class can only learn patterns which are learnable by the elements of that class. ✓

d. The VC-dimension of the hypothesis class corresponding to the AdaBoost can grow faster in the number of weak learners, T, than the VC-dimension of the weak learners, d.

1 - Yes

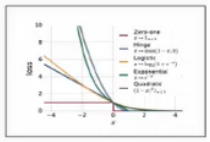
The loss function optimized by AdaBoost

The viewpoint of loss function optimization suggest alternative boosting algorithms by changing the loss function

- For example, LogitBoost (Friedman et al. 2000) minimizes the logistic loss:

$$F_{logit}(\alpha) = \sum_{i=1}^{m} \log(1+e^{-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)})$$

- LogitBoost is closely related logistic regression (LR): if we consider the weak learners h_t as the input features and α_t as the feature weights, LogitBoost is essentially learning a logistic regression model (except for some constants that differ from LR).



Friedman, J., Hastie, T., Tibshirani, R. (2000). "Additive logistic regression: a statistical view of boosting". Annals of Statistics. 28 (2): 337–407

2 - Yes

AdaBoost

For the round $t+1$, the weights on the training examples are re-weighted:

$$D_{t+1}(i) = D_t(i) \cdot \frac{e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

- The term $y_i \alpha_t h_t(\mathbf{x})$ can be seen as a margin of example (weighted by α_t)
- Feeding the margin through an negative exponential causes an exponential up-weighting of misclassified examples (those with negative margin) and down-weighting of correctly classified examples (positive margin)
- The factor: $Z_t = \sum_{i=1}^{m} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$ is a normalization factor ensuring that D_t sums up to 1.

The sequence of the re-weighting is that the weak learners for subsequent rounds will focus on the examples that were misclassified ⇒ increases diversity

3 - Not

Why can an ensemble work in practice?

Representational reason (Dietterich, 2000):

- The representational capacity of a hypothesis space can be extended beyond the search space by averaging over several hypotheses.
- For example: combination of hyperplanes let us represent sets of convex polygons, which is more general than sets of half-spaces, represented by single hyperplanes
- Thus an ensemble can learn patterns outside the original hypothesis class $\mathcal{H}$.

Dietterich, Thomas G. "Ensemble methods in machine learning." International workshop on multiple classifier systems. Springer, Berlin, Heidelberg, 2000.

4 - Yes

VC-dimension of AdaBoost

- The hypothesis class of AdaBoost after T rounds is given by

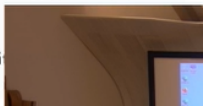
$$\mathcal{F}_T = \left\{ sgn\left(\sum_{t=1}^{T} \alpha_t h_t\right) : \alpha_t > 0, h_t \in H \right\}$$

- The VC-dimension of $\mathcal{F}_T$ can be bounded in terms of the VC-dimension d of the weak learners:

$$VCdim(\mathcal{F}_T) \leq 2(d+1)(T+1)\log_2((T+1)e)$$

which grows as $O(dT \log T)$

- This suggests that AdaBoost could overfi
however empirically it has been observed



Question **4**

Flag question Mark 0.0 out of 2.0 Incorrect

The performance of four different classifiers mentioned in the lectures are compared in this question. In the implementation, this example of the Sklearn package can be used as a prototype of this kind of tasks.

In our problem, we have three data sets:

```
from sklearn.datasets import load_breast_cancer

from sklearn.datasets make_moons

from sklearn.datasets make_circles
```

The dataset loading functions are parameterized in this way:

```
load_breast_cancer(return_X_y=True),  ## X input, y output

make_moons(n_samples = 100, noise=0.3, random_state=1),

make_circles(n_samples = 100, noise=0.2, factor=0.5, random_state=1)
```

The four learners are the following ones:

```
from sklearn.neural_network import MLPClassifier

from sklearn.ensemble import AdaBoostClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import RandomForestClassifier
```

Those learners need to be parameterized in this way to receive comparable results:

```
MLPClassifier(hidden_layer_sizes = (100,), alpha=0.0001, max_iter=200, random_state=1),

AdaBoostClassifier(n_estimators = 100, random_state=1)

GradientBoostingClassifier(n_estimators=100, learning_rate=1, max_depth=1, random_state=1)

RandomForestClassifier(max_depth=4, random_state=1)
```

The data sets are cut only once into training and test following the Sklearn example code, but the test size is selected as 0.5, and the random state is set to 1.

The task is to report that classifier which provides the best result on these three datasets. In the comparison, the performance of the classifiers is measured by the average of the *Area Under the Receiver Operating Characteristic Curve (ROC AUC)* computed on each datasets separately.

Hint: You could use and modify the Sklearn example, and all available functions provided by that package. Be careful, the example code also contains several details of the visualization that is not important to answer this question.

Select one:

- a. MLPClassifier ✗
- b. AdaBoostClassifier
- c. GradientBoostingClassifier
- d. RandomForestClassifier

(1) MLPClassifier
(2) AdaBoostClassifier
(3) GradientBoostingClassifier
(4) *** RandomForestClassifier

Solution:

Correct answer is RandomForestClassifier

c:\Users\nguye\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\neural_network_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
warnings.warn(

Neural Net 0.888536221060493
AdaBoost 0.9370799103808813
Gradient Boosting 0.9474287848074257
Random Forest 0.9315854048863758

Gradient Boosting 0.3158
AdaBoost 0.3124
Random Forest 0.3105
Neural Net 0.2962

c:\Users\nguye\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\neural_network_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
warnings.warn(

Neural Net 0.96
AdaBoost 0.8199999999999998
Gradient Boosting 0.8199999999999998
Random Forest 0.9600000000000001

Random Forest 0.6305
Neural Net 0.6162
Gradient Boosting 0.5891
AdaBoost 0.5857

c:\Users\nguye\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\neural_network_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
warnings.warn(

Neural Net 0.88
AdaBoost 0.76
Gradient Boosting 0.7799999999999999
Random Forest 0.9

Random Forest 0.9305
Neural Net 0.9095
Gradient Boosting 0.8491
AdaBoost 0.8390

Quiz 6

Question **1**

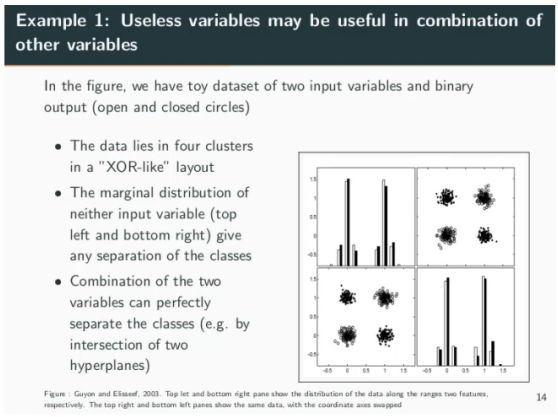
Flag question Mark 1.00 out of 1.00 Correct

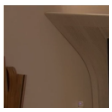
Select the true statements *Note: Multiple choices may be correct. Marks are not awarded for partially correct solutions*

Select one or more:

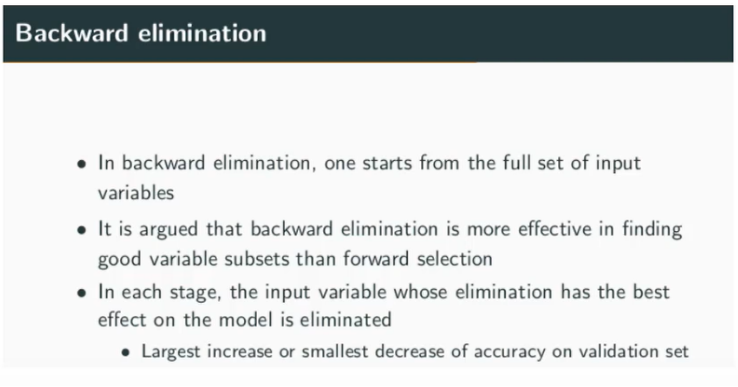
- a. Selecting a subset of features most correlated to the output values is guaranteed to find the most useful subset of features for the supervised learning.
- b. Backward elimination can be expected to produce better results than forward selection. ✓
- c. If data contains highly correlated variables, l_1-norm regularization on them might result in only one of the corresponding coefficients being non-zero. In other words, if $x_i = x_j$ and $w_i = 1$, $w_j = 0$ is a valid solution for the lasso problem, then any $w_i = \alpha$, $w_j = 1 - \alpha$ with $0 \leq \alpha \leq 1$ is also. ✓
- d. l_∞-norm is the limit of l_p norm, defined as $l_\infty(x) = \max(|x_1|, |x_2|, \ldots, |x_d|)$. Like l_0 or l_1-norms, the l_∞ promotes sparsity.

A Selecting a subset of features most correlated to the output values is guaranteed to find the most useful subset of features for the supervised learning task. **False**





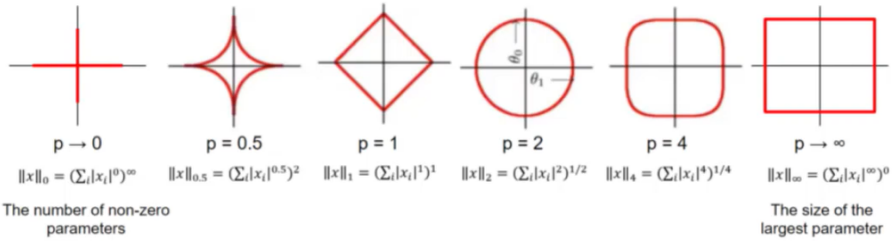
B Backward elimination can be expected to produce better results than forward selection. **True**



C If data contains highly correlated variables, l_1-norm regularization on them might result in only one of the corresponding coefficients being non-zero. In other words, if $x_i = x_j$ and $w_i = 1$, $w_j = 0$ is a valid solution for the lasso problem, then any $w_i = \alpha$, $w_j = 1 - \alpha$ with $0 \leq \alpha \leq 1$ is also. **True**

▶ True; from highly similar features it is possible that Lasso will select only one of them, since adding both won't add power to the predictive model

D l_∞-norm is the limit of l_p norm, defined as $l_\infty(x) = \max(|x_1|, |x_2|, ..., |x_d|)$. Like l_0 or l_1-norms, the l_∞ promotes sparsity. **False**



Information

Flag question

Consider the code template loading the breast cancer Wisconsin data from sklearn . Investigate various feature transformation techniques: centering, standardisation, unit range and normalisation with l_2-norm as defined one the slides "Examples of feature transformations". Fit Lasso from sklearn (use **tol=1e-3** and **max_iter=1e5**) to the transformed training sets and investigate the accuracies and sparsities obtained with different levels of regularisation. In the sklearn's implementation the regularisation parameter, called λ in the lecture slides, is set with **alpha**.

Note: while feature transformation techniques can be (and often are) applied consecutively, in this question only one feature transformation technique is applied at a time. Also recall good machine learning practices about testing: nothing about the test set should be leaked to the training stage. This means that the feature transformations should be also defined only based on the training data.

Hints: The following numpy functions might be useful (but by no means mandatory): count_nonzero , logspace .
If you obtain multiple results with the same level of sparsity, you can consider only the results with maximum accuracy.

Question **2**

Flag question Mark 0.00 out of 0.50 Incorrect

With which feature transformation technique(s) can you obtain at least 90% accuracy on the test set with fewest selected features? *Note: Multiple choices may be correct. Marks are not awarded for partially correct solutions*

Select one or more:

- ☑ a. No feature transformation ✗
- ☑ b. Centering ✗
- ☑ c. Standardisation ✗
- ☐ d. Unit range
- ☐ e. Normalization of feature vectors

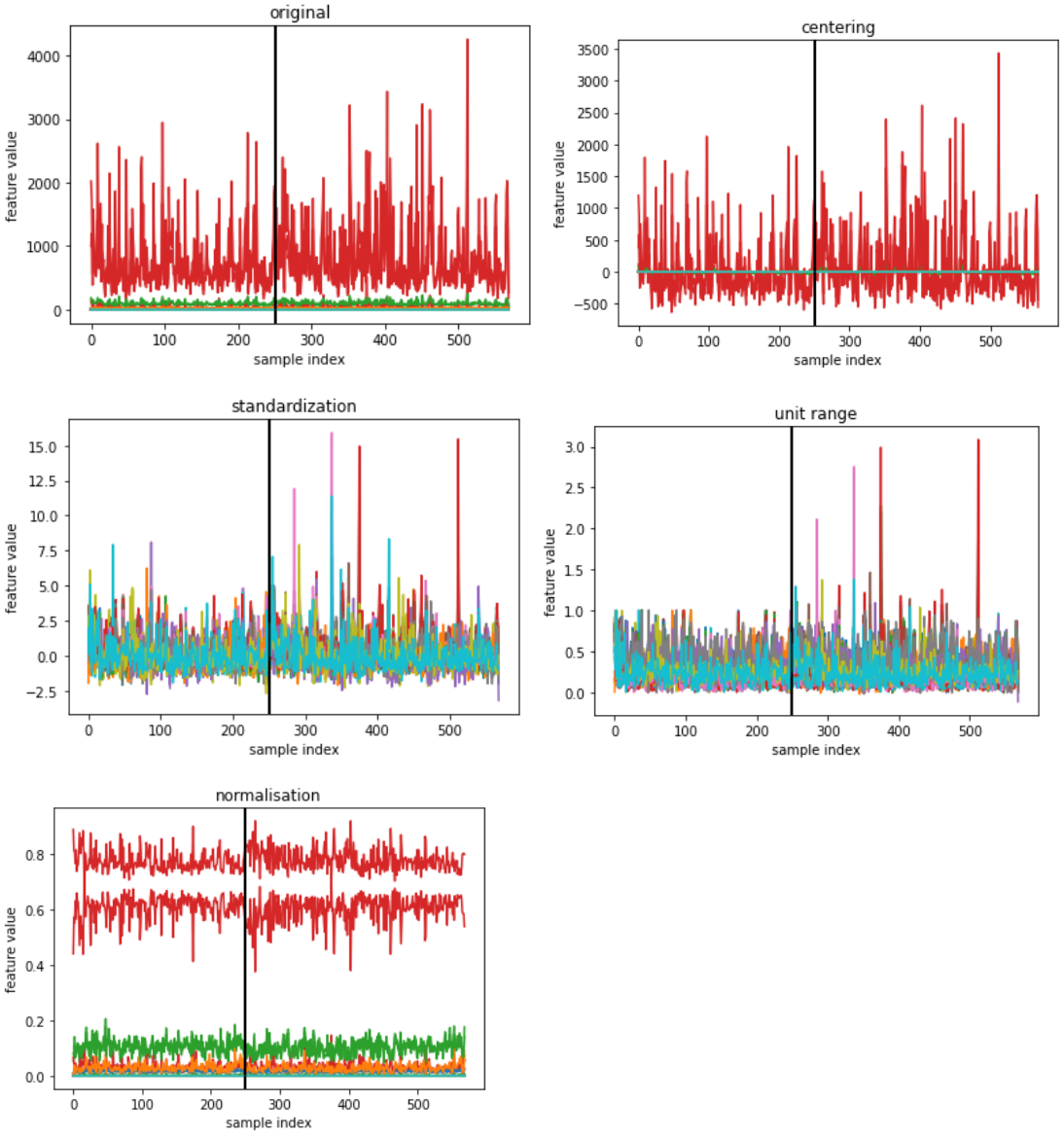
A No feature transformation

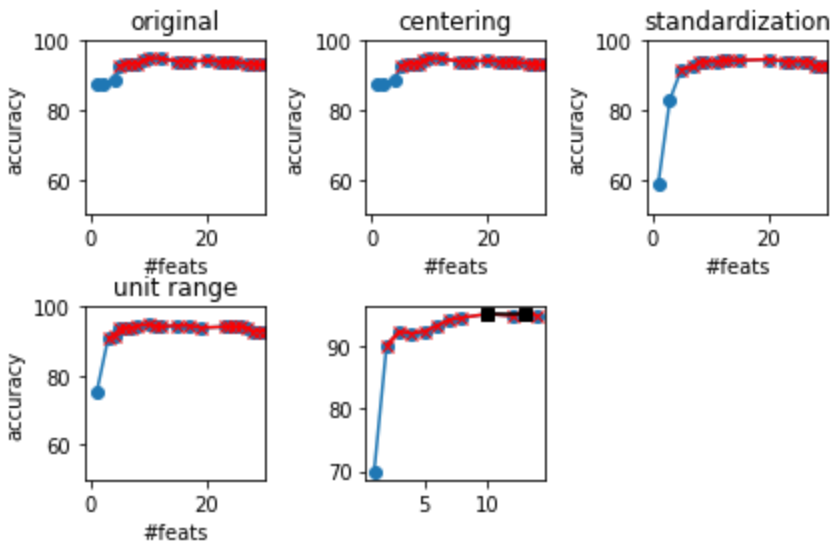
B Centering

C Standardisation

D Unit range

E Normalization of feature vectors **Correct**





Question **3**

Flag question Mark 0.00 out of 0.50 Incorrect

With which feature transformation technique(s) can you obtain over 85% accuracy on the test set already with one feature? *Note: Multiple choices may be correct. Marks are not awarded for partially correct solutions*

Select one or more:

- a. No feature transformation
- b. Centering
- c. Standarisation ✗
- d. Unit range
- e. Normalization of feature vectors

A No feature transformation **Correct**

B Centering **Correct**

C Standardisation

D Unit range

E Normalization of feature vectors

Question **4**

Flag question Mark 0.00 out of 1.00 Incorrect

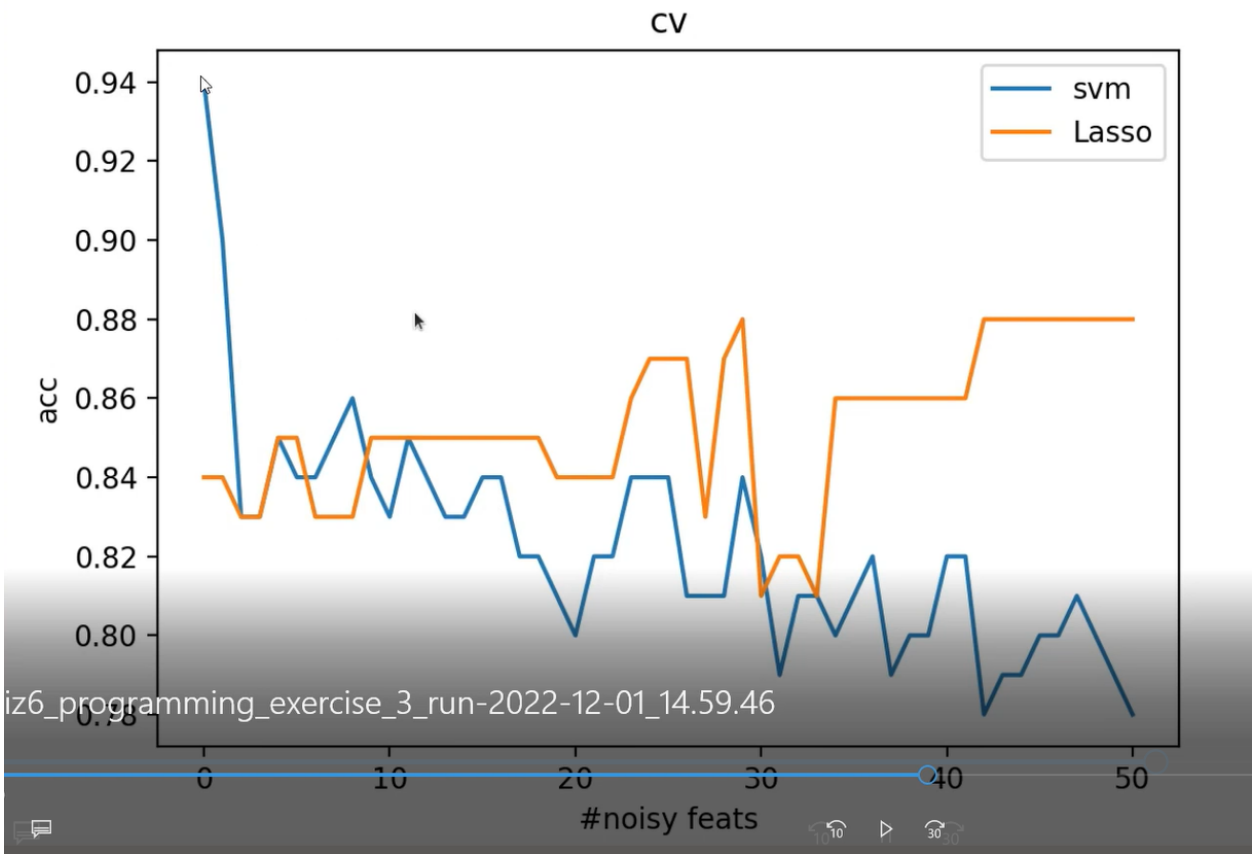
Consider the code template sampling the two moons toy dataset from sklearn . Investigate the effect of adding noisy features (as given in the template) to the original data with support vector machine using RBF kernel, and Lasso imported in the template. For all noise amounts, perform 5-fold cross-validation to select the optimal parameters from the lists of parameters given in template (for all other parameters use the defaults) with a grid search. See slide "n-Fold Cross-Validation" from lecture 4 to recall the procedure; do not shuffle the data here but consider them in order as illustrated there. Use accuracy as the metric. If ties occur during the CV, select the parameters with lowest index.

Hints: The following numpy functions and sklearn modules might be useful (but by no means mandatory): argmax , mean , KFold .

What is the lowest amount of noisy features added to the original data, when Lasso accuracy on test set is better than SVM?

Answer: 10 ✗

Correct answer is 5



```
0
best lasso parameter is: 0.1
best gamma parameter is: 1
best svm parameter is: 1
1
best lasso parameter is: 0.1
best gamma parameter is: 1
best svm parameter is: 10
2
best lasso parameter is: 0.1
best gamma parameter is: 0.001
best svm parameter is: 100
3
best lasso parameter is: 0.01
best gamma parameter is: 0.001
best svm parameter is: 100
4
best lasso parameter is: 0.01
best gamma parameter is: 0.01
best svm parameter is: 1
5
best lasso parameter is: 0.01
best gamma parameter is: 0.001
best svm parameter is: 10
```

Question **5**

Flag question Mark 0.00 out of 1.00 Incorrect

Select the true statements *Note: Multiple choices may be correct. Marks are not awarded for partially correct solutions*

Select one or more:

- a. Both OVA and OVO can be seen as special cases of ECOC approach ✗
- b. OVA scheme always produces the optimal error rate for a classifier
- c. One-vs-one is more robust to uneven classes than one-vs-all ✓
- d. Comparing classifier scores in one-vs-all scheme is a meaningful way to break the possible ties
- e. One-vs-one model is more prone to overfitting than one-vs-all ✓

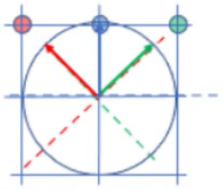
A Both OVA and OVO can be seen as special cases of ECOC approach **False**

- OVO strategy considers only pairs of classes
- In ECOC all the binary classification problems consider all the classes

B OVA scheme always produces the optimal error rate for a classifier **False**

Example: sub-optimality of OVA classification

- However, the three classes are separable by three hyperplanes $f_\ell(\mathbf{x}) = \mathbf{w}_\ell^T \mathbf{x}$, $\mathbf{w}_{red} = (-1/\sqrt{2}, 1/\sqrt{2}), \mathbf{w}_{blue} = (0, 1)$ and $\mathbf{w}_{green} = (1/\sqrt{2}, 1/\sqrt{2})$ using the rule $h(\mathbf{x}) = \text{argmax}_\ell f_\ell(\mathbf{x})$
- Note that the hyperplane $\mathbf{w}_{blue}$ is not a good classifier as a independent model, its empirical error rate is 80%!



optimal empirical error rate: 0%

- Thus we see that independent training of the binary hypotheses loses information and may result in sub-optimal error rates.

C One-vs-one is more robust to uneven classes than one-vs-all **True**

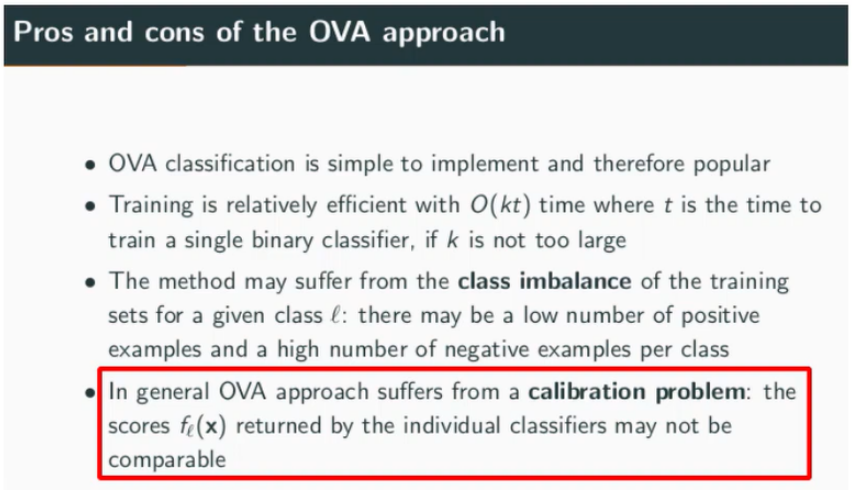
Pros and cons of the OVA approach

- OVA classification is simple to implement and therefore popular
- Training is relatively efficient with $O(kt)$ time where t is the time to train a single binary classifier, if k is not too large
- The method may suffer from the **class imbalance** of the training sets for a given class ℓ: there may be a low number of positive examples and a high number of negative examples per class

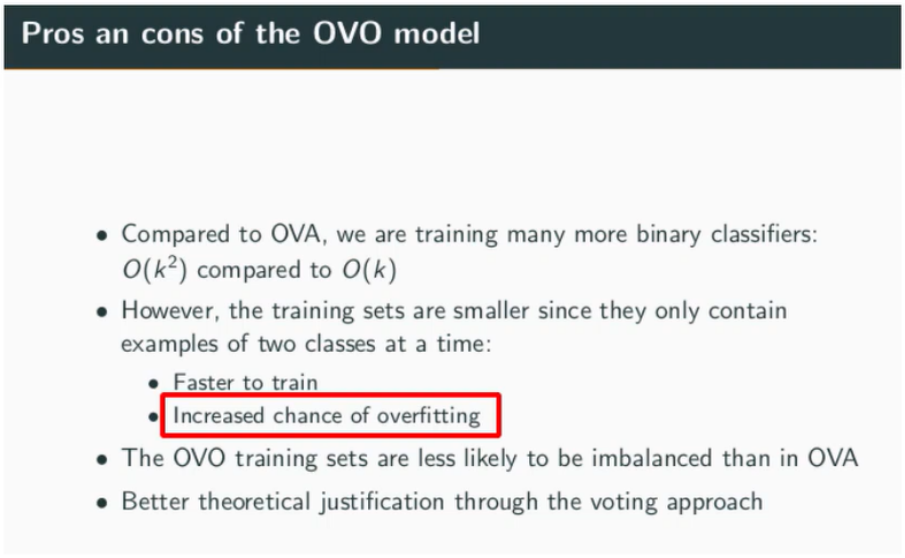
Pros an cons of the OVO model

- Compared to OVA, we are training many more binary classifiers: $O(k^2)$ compared to $O(k)$
- However, the training sets are smaller since they only contain examples of two classes at a time:
  - Faster to train
  - Increased chance of overfitting
- The OVO training sets are less likely to be imbalanced than in OVA

D Comparing classifier scores in one-vs-all scheme is a meaningful way to break the possible ties **False**



E One-vs-one model is more prone to overfitting than one-vs-all **True**



Information

Flag question

Consider the ECOC strategy for classifying cats, dogs, parrots, axolotls, kiwis and pangolins. What are the minimum and the maximum lengths of the codewords? Redundant codes should not be included for maximum length.

Question **6**

Flag question Mark 0.50 out of 0.50 Correct

Minimum length:

Answer: 3 ✓

Question **7**

Flag question Mark 0.50 out of 0.50 Correct

Maximum length (without redundant codes):

Answer: 31 ✓

Question 5

Error-correcting codes (ECOC)

- Error-correcting output codes (ECOC) is a general methods for reducing multi-class problems to binary classification
- In the ECOC approach, each class ℓ is allocated a codeword m_ℓ of length $c > 1$
- In the simplest case a binary vector can be used $m_\ell \in \{-1, +1\}^c$
- The code words of all k classes together form a matrix $M \in \{-1, +1\}^{k \times c}$

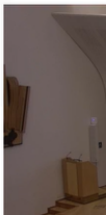
codes

classes	1	2	3	4	5	6
1	-1	-1	-1	+1	-1	-1
2	+1	-1	-1	-1	-1	-1
3	-1	+1	+1	-1	+1	-1
4	+1	+1	-1	-1	-1	-1
5	+1	+1	-1	-1	+1	-1
6	-1	-1	+1	+1	-1	+1
7	-1	-1	+1	-1	-1	-1
8	-1	+1	-1	+1	-1	-1

$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
-1	+1	+1	-1	+1	+1

new example x

- codes: the columns
- codewords: the rows



Minimum length:

- How many bits are needed to form k different codes?
- How many different l-length binary codes exist: 2^l.
- $\log_b(b^x) = x$
- Require $k = 6$ unique codes
- $\log_2(k) = \log_2(6) = 2.58 \Rightarrow 3$

Maximum length:

- For k classes it is possible to write 2^k different binary codes of length k.
- However half of the codes are redundant! For four classes codes [-1, -1, 1, 1] and [1, 1, -1, -1] describe the same decision boundary.
- Also, a code that classifies everything to the same class (i.e. every sample is inside/outside decision boundary) is not meaningful for ECOC
- $2^k/2 - 1 = 2^{6-1} - 1 = 2^5 - 1 = 32 - 1 = 31$
  - Visualisations of exhaustive codes with $k = 3$ and $k = 5$

Table 5: All possible columns for a three-class problem. Note that the last four columns are complements of the first four and that the first column does not discriminate among any of the classes.

	Code Word							
Class	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
c_0	0	0	0	0	1	1	1	1
c_1	0	0	1	1	0	0	1	1
c_2	0	1	0	1	0	1	0	1

$$2^{k-1} - 1 = 2^2 - 1 = 3$$

Table 6: Exhaustive code for k=5.

Row	Column														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1
4	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

$$2^{k-1} - 1 = 2^4 - 1 = 15$$

Images from: Dietterich, T. G., & Bakiri, G. (1994). Solving multiclass learning proble error-correcting output codes. Journal of artificial intelligence research 2

Final quiz

Question **1**

Flag question Marked out of 5.00 Not yet answered

Which assumption is correct in a general learning setup:

- a. The learner has access to the the function that generates the true labels, but not to the underlying distribution of the data
- b. The learner does not know the underlying distribution of the data or the function generating the true labels
- c. The empirical error computation needs to have access to the underlying distribution generating the data

Clear my choice

a) wrong. If we already know the function that generates the true labels then there is no use for the learning at all
b) This is the main reason for using machine learning: predict data without knowing true function nor data distribution
c) No, distribution of data is not required to calculate the empirical error

Question **2**

Flag question Marked out of 5.00 Not yet answered

Consider a consistent classifier with hypothesis set size $|\mathcal{H}| = 52$ and sample size m. How many samples are needed to keep the generalization error the same, if the confidence level is changed by changing δ from 0.05 to 0.1?

- a. $0.63m$
- b. $0.90m$
- c. $1.11m$
- d. $2.07m$

Clear my choice

Finite hypothesis class - consistent case

- Sample complexity bound relying on the size of the hypothesis class (Mohri et al, 2018): $Pr(R(h_s) \leq \epsilon) \geq 1 - \delta$ if

$$m \geq \frac{1}{\epsilon}(\log(|\mathcal{H}|) + \log(\frac{1}{\delta}))$$

- An equivalent generalization error bound:

$$R(h) \leq \frac{1}{m}(\log(|\mathcal{H}|) + \log(\frac{1}{\delta}))$$

- Holds for any finite hypothesis class assuming there is a consistent hypothesis, one with zero empirical risk
- Extra term compared to the rectangle learning example is the term $\frac{1}{\epsilon}(\log(|\mathcal{H}|))$
- The more hypotheses there are in $\mathcal{H}$, the more training examples are needed

=> 1/m1 (log(52) + log(1/0.05)) = 1/m2 (log(52)+log(1/0.1)) => m2 = 0.9m1

Question **3**

Flag question Marked out of 5.00 Not yet answered

Which of the following claims is true?

- a. Rademacher complexity depends on the distribution generating the data
- b. Rademacher complexity measures the performance of the learning algorithm in the worst-case scenario of assigning labels to samples in adversarial way
- c. It is not possible to estimate Rademacher complexity empirically

Clear my choice

Rademacher vs. VC

Note the differences between Rademacher complexity and VC dimension

- VC dimension is independent of any training sample or distribution generating the data: it measures the worst-case where the data is generated in a bad way for the learner
- Rademacher complexity depends on the training sample thus is dependent on the data generating distribution
- VC dimension focuses the extreme case of realizing all labelings of the data
- Rademacher complexity measures smoothly the ability to realize random labelings

Question **4**

Flag question Marked out of 5.00 Not yet answered

Generalization error of a hypothesis class can be partitioned as

$$R(h) = R^* + \epsilon_{estimation} + \epsilon_{approximation}$$

into Bayes error R^*, approximation error $\epsilon_{approximation}$ (also known as bias) and estimation error $\epsilon_{estimation}$ (also known as variance).
Which of the following claims is true?

- a. The Bayes error term R^* cannot be reduced
- b. The Bayes error term R^* can be reduced by choosing a good hypothesis class $\mathcal{H}$
- c. $\epsilon_{approximation}$ can be reduced by using more training data

Clear my choice

a) True, the Bayes error is the theoretical lowest achievable error => R* cannot be reduced
b) False, this is for approximation error: reducing it needs better hypothesis H
c) False, this is for estimation error: reducing it needs more training data

Question **5**

Flag question Marked out of 5.00 Not yet answered

Let x be chosen from the interval $[-1, +1]$, and the labels, y, from the set $\{0, 1\}$. we are given the following conditional probabilities for all x and y:

$$pr(1|x) = \begin{cases} +x+1 & x \in [-1, 0] \\ -x+1 & x \in [0, +1] \end{cases}$$
$$pr(0|x) = \begin{cases} -x & x \in [-1, 0] \\ +x & x \in [0, +1] \end{cases}$$

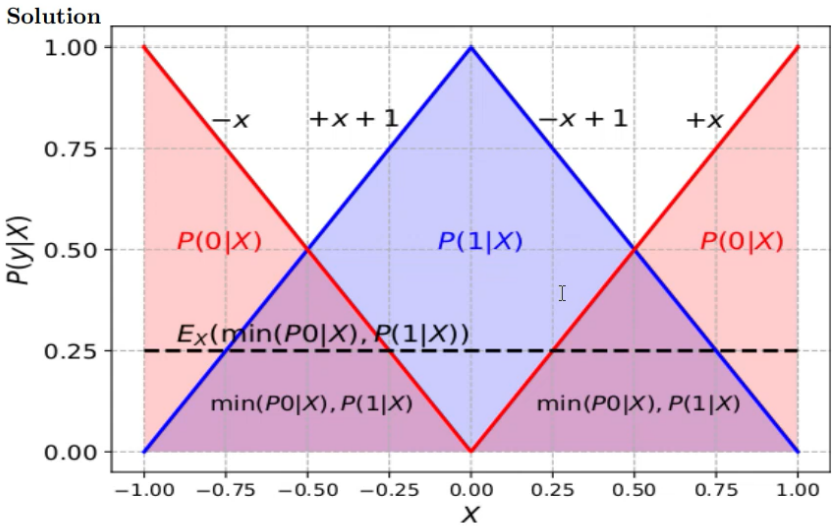
What is the Bayes error of the Bayes classifier relating to this model if x is uniformly distributed on $[-1, +1]$?

Hint: you might solve the problem by drawing a plot representing the functions of the conditional probabilities.

- a. 0.75
- b. 0.5
- c. 0.3
- d. 0.25

Clear my choice

Solution



$$\text{noise}(x) = \min(P(0|x), P(1|x)) = \begin{cases} +x+1 & x \in [-1, -0.5] \\ -x+0 & x \in (-0.5, 0] \\ +x+0 & x \in [0, 0.5] \\ -x+1 & x \in (0.5, 1] \end{cases}$$

If the distribution of x is uniform, then the density of x, $f(x) = 0.5$, therefore we have

$$\mathrm{E}_x(\text{noise}(x)) = \int_{-1}^{+1} \text{noise}(x) f(x) dx$$
$$= \int_{-1}^{-0.5} 0.5(x+1)\, dx + \int_{-0.5}^{0} 0.5(-x)\, dx + \int_{0}^{0.5} 0.5x\, dx + \int_{0.5}^{1} 0.5(-x+1)\, dx = \boxed{0.25}.$$

1

Question **6**

Flag question Marked out of 5.00 Not yet answered

Let us consider the perceptron algorithm and a linearly separable training set. Novikoff's theorem states an upper bound on the number of iterations required that for every training example, $(\mathbf{x}_i, y_i)$ satisfies $y_i \mathbf{w}^T \mathbf{x}_i \geq 0$. How does that bound depend on the largest achievable geometric margin?

- a. It is independent from that margin.
- b. Increases when the margin increases.
- c. Decreases when the margin increases.

Clear my choice

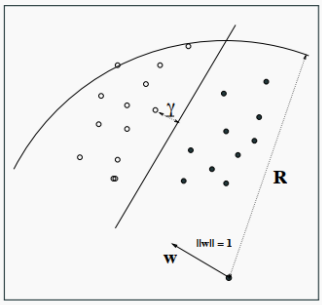
Convergence of the perceptron algorithm

- The perceptron algorithm can be shown to eventually converge to a consistent hyperplane if the two classes are **linearly separable**, that is, if there exists a hyperplane that separates the two classes
- Theorem (Novikoff):
  - Let $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ be a linearly separable training set.
  - Let $R = \max_{\mathbf{x}_i \in S} \|\mathbf{x}_i\|$.
  - Let there exist a vector $\mathbf{w}_*$ that satisfies $\|\mathbf{w}_*\| = 1$ and $y_i \mathbf{w}_*^T \mathbf{x}_i + b_{opt} \geq \gamma$ for $i = 1 \ldots, m$.
  - Then the perceptron algorithm will stop after at most $t \leq (\frac{2R}{\gamma})^2$ iterations and output a weight vector $\mathbf{w}^{(t)}$ for which $y_i \mathbf{w}^{(t)} \mathbf{x}_i \geq 0$ for all $i = 1 \ldots, m$

Convergence of the perceptron algorithm

The number of iterations in the bound $t \leq (\frac{2R}{\gamma})^2$ depend on:

- γ: The largest achievable geometric margin so that all training examples have at least that margin
- R: The smallest radius of the d-dimensional ball that encloses the training data
- Intuitively: how large the margin in is relative to the distances of the training points



Question **7**

Flag question Marked out of 5.00 Not yet answered

For a given sample of inputs and outputs, $\{\mathbf{x}_i, y_i\}$, $i = 1, \ldots, m$, in the dual of the soft-margin SVM, the normal vector of the separating hyperplane, $\mathbf{w}$, can be expressed as $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$, where $\{\alpha_i\}$ are the optimal dual variables. Which of these statement is true?

a. α_i is greater than 0 for all i

b. For a given i, α_i greater than 0 if $y_i = 1$

c. For a given i, α_i is greater than 0 if $\mathbf{x}_i$ has functional margin smaller or equal to 1.

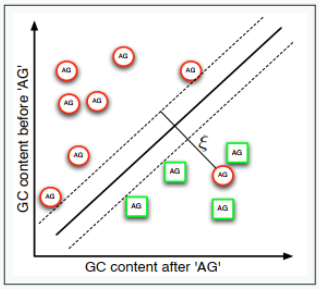
Clear my choice

Dual representation of the optimal hyperplane

It can be shown theoretically that the **optimal hyperplane** of the soft-margin SVM has a **dual representation** as the linear combination of the training data

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

- The coefficients, also called the **dual variables** are non-negative $\alpha_i \geq 0$
- The positive coefficients $\alpha_i > 0$ appear if and only if $\mathbf{x}_i$ is a support vector, for other training points we have $\alpha_i = 0$



The soft-margin SVM allows non-separable data by using the relaxed the margin constraints

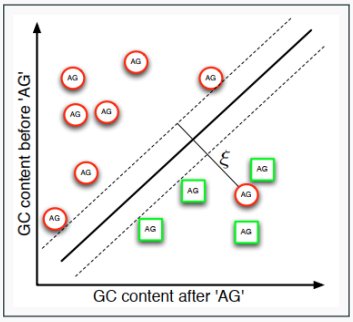
$$Minimize \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i$$

$$w.r.t \text{ variables } \mathbf{w}, \boldsymbol{\xi}$$

$$\text{Subject to } y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i$$

$$\text{for all } i = 1, \ldots, m.$$

$$\xi_i \geq 0, \text{for all } i = 1, \ldots, m.$$



Question **8**

Flag question Marked out of 5.00 Not yet answered

Let the feature embedding $\phi(\mathbf{x})$ underlying the polynomial kernel, $\kappa_{pol}(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z} + c)^q$, be represented explicitly as a vector. What is the dimension of the explicit feature vector for polynomial kernel of degree $q = 3$ when inputs satisfy $\mathbf{x} \in \mathbb{R}^3$.

- (•) a. 20
- ( ) b. 10
- ( ) c. 15

Clear my choice

Example: Polynomial kernel on 2D inputs

- Consider two-dimensional inputs $\mathbf{x} = [x_1, x_2]^T \in \mathbb{R}^2$
- The second degree polynomial kernel is given by $\kappa(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^2$
- We can write it as a inner product in $\mathbb{R}^6$:

$$\begin{aligned}\kappa(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^2 &= (x_1x_1' + x_2x_2' + c)^2 = \\ &= x_1x_1'x_1x_1' + x_2x_2'x_2x_2' + c^2 + \\ &+ 2x_1x_1'x_2x_2' + 2cx_1x_1' + 2cx_2x_2' \\ &= \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2c}x_1 \\ \sqrt{2c}x_2 \\ c \end{bmatrix}^T \begin{bmatrix} x_1'^2 \\ x_2'^2 \\ \sqrt{2}x_1'x_2' \\ \sqrt{2c}x_1' \\ \sqrt{2c}x_2' \\ c \end{bmatrix} \\ &= \phi(\mathbf{x})^T\phi(\mathbf{x}'),\end{aligned}$$

where $\phi(\mathbf{x}) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2c}x_1, \sqrt{2c}x_2, c]^T$

When we expand, we have (x1z1 + x2z2 + x3z3 + c)^3
The result is 20 terms in the expansion, which is the explicit feature vector size
https://www.symbolab.com/solver/expand-calculator/expand%20%5Cleft(x_%7B1%7Dz_%7B1%7D%2Bx_%7B2%7Dz_%7B2%7D%2Bx_%7B3%7Dz_%7B3%7D%2Bc%5Cright)%5E%7B3%7D?or=input

Question **9**

Flag question Marked out of 5.00 Not yet answered

Which of the following statements about multi-layer perceptrons (MLP) is true?

- a. Stochastic gradient descent is guaranteed to find the parameters for a MLP that minimize the zero-one error on training data.
- b. MLPs can be used to represent arbitrary boolean functions
- c. Using a linear activation function in each neuron in each layer results in a MLP that computes a non-linear function of the inputs.

Clear my choice

Why do we need non-linear activation functions?

- Consider having two layer network with first layer computing $z_h = \sum_j w_{hj} x_j$ and the second layer computing $y_i = \sum_j v_{ih} z_h$
- The total function is thus:

$$y_i = \sum_h v_{ih} \sum_j w_{hj} x_j = \sum_j \sum_h v_{ih} w_{hj} x_j$$

- We can compute the same with a linear function:

$$y_i = \sum_j u_{ij} x_j$$

where $u_{ij} = \sum_h v_{ih} w_{hj}$

- Thus there is no real non-linearity in the model and our model reduces to learning a linear hyperplane

To make the network structure useful, we need non-linear activation functions

The bad news: hardness of training MLPs

Learning optimal weights for MLPs and other neural networks is computationally hard (Shalev-Shwartz and Ben-David, 2014):

- It is NP-hard to find the parameters that minimizes the empirical error, for a network with a single hidden layer that contains 4 neurons or more
- Even close-to-minimal error is NP-hard to achieve
- Changing the structure of the network is not likely to make learning easier, since any function class that can represent intersections of halfspaces is NP-hard under some cryptographic assumptions

Thus in practice we need to resort in heuristic optimization approaches with no theoretical guarantees of optimality

Representing arbitrary boolean functions with neural nets

- Perceptron can represent all three basic logical operators AND, OR and NOT
- All Boolean functions can be represented by combinations of these basic operations
- Thus, MLPs can in principle represent arbitrary Boolean functions
- However, **learning** arbitrary Boolean functions may still require prohibitive amount of data and time (e.g. the VC dimension of arbitrary Boolean functions of d variables is 2^d)

Question **10**

Flag question Marked out of 5.00 Not yet answered

Assume 17 base learners with independent true risk 0.45 ($\epsilon = 0.45$), are grouped to form a strong learner via majority voting. What is the probability of having an incorrect aggregated prediction?

- a. 0.3
- b. 0.34
- c. 0.18
- d. 0.25

Clear my choice

In order to get an incorrect aggregated prediction, at least 9/17 trees must make the wrong prediction. We can use the cumulative distribution function of the Bernoulli distribution with parameter theta = 0.45 to calculate the probability:

- Enter a value in each of the first three text boxes (the unshaded boxes).
- Click the **Calculate** button to compute binomial and cumulative probabilities.

Probability of success on a trial	0.45
Number of trials	17
Number of successes (x)	9
Binomial probability: P(X=9)	0.15403
Cumulative probability: P(X<9)	0.66256
Cumulative probability: P(X≤9)	0.81659
Cumulative probability: P(X>9)	0.18341
Cumulative probability: P(X≥9)	0.33744

Calculate

The answer is 0.337 approx 0.34

Question **11**

Flag question Marked out of 5.00 Not yet answered

In each iteration of AdaBoost, the weights of the examples are modified to increase the weights on the mistakes made by :

- a. The last weak learner
- b. The ensemble in the previous iteration
- c. The ensemble including the last weak learner

Clear my choice

AdaBoost$(S = ((x_1, y_1), \ldots, (x_m, y_m)))$

1 **for** $i \leftarrow 1$ **to** m **do**
2 $\mathcal{D}_1(i) \leftarrow \frac{1}{m}$
3 **for** $t \leftarrow 1$ **to** T **do**
4 $h_t \leftarrow$ base classifier in $\mathcal{H}$ with small error $\epsilon_t = \mathbb{P}_{i \sim \mathcal{D}_t}\left[h_t(x_i) \neq y_i\right]$
5 $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
6 $Z_t \leftarrow 2\left[\epsilon_t(1-\epsilon_t)\right]^{\frac{1}{2}}$ ▷ normalization factor
7 **for** $i \leftarrow 1$ **to** m **do**
8 $\mathcal{D}_{t+1}(i) \leftarrow \frac{\mathcal{D}_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
9 $f \leftarrow \sum_{t=1}^{T} \alpha_t h_t$
10 **return** f

Question **12**

Flag question Marked out of 5.00 Not yet answered

Which option is true?

- a. For feature selection, backward elimination is computationally not as demanding as forward selection.
- b. The performance of a machine learning algorithm can vary a lot based on the feature transformation applied to the data
- c. While l_1 norm regularization induces sparsity in the learning problem, computationally the machine learning problem is not convex.

Clear my choice

a) false. backward elimination is computationally demanding as forward selection
b) true. There are radical ML performance under different data transformation
c) false. L1 induced both sparsity in the learning problem and it is also convex

Question **13**

Flag question Marked out of 5.00 Not yet answered

With the following weight matrix $\mathbf{W}$ for a multi-class SVM, to which class (numbering starts from 0) will a new sample $x = (-2, 1)$ be classified into?

$$\mathbf{W} = \begin{bmatrix} 2 & -3 & 0.5 & 1 & -2.5 \\ 0 & 0.5 & -1 & -2 & 1 \end{bmatrix}$$

- a. 0
- b. 1
- c. 2
- d. 3
- e. 4

Clear my choice

$$\begin{pmatrix} 2 & -3 & 0.5 & 1 & -2.5 \\ 0 & 0.5 & -1 & -2 & 1 \end{pmatrix} T \begin{pmatrix} -2 \\ 1 \end{pmatrix} = \begin{pmatrix} -4 \\ 6.5 \\ -2 \\ -4 \\ 6 \end{pmatrix}$$

Highest value result is 6.5, which corresponds to feature number 1

Question **14**

Flag question Marked out of 5.00 Not yet answered

Alice and Bob wish to have a literature circle, and consider Alice's adventures in Wonderland (AAW) , Diskworld series (DW , Dune, Foundation series and Pride and Prejudice (PP).
Alice prefers the order $AAW \succ DW \succ PP \succ Dune \succ Foundation$ while
Bob prefers the order $PP \succ Dune \succ AAW \succ DW \succ Foundation$.

To resolve the conflict they decide that they should read the books in order σ that has a small value $d_{max}(\sigma) = \max d_K(\sigma_{Alice}, \sigma), d_K(\sigma_{Bob}, \sigma)$ where d_K is the Kendall's distance, meaning that σ is close to both Alice's and Bob's preferred order.

Which of the following orders would be the best for Alice and Bob:

- a. $PP \succ DW \succ Dune \succ AAW \succ Foundation$
- b. $AAW \succ PP \succ DW \succ Dune \succ Foundation$
- c. $PP \succ AAW \succ DW \succ Dune \succ Foundation$

Clear my choice

a) Kendall(Alice) = 4, Kendall(Bob) = 2 => max = 4
b) Kendall(Alice) 1, Kendall(Bob) = 3 => max = 3
c) Kendall(Alice) = 2, Kendall(Bob) = 2 => max = 2