Assignment 2 - Machine Learning

Caio de Próspero Iglesias

January 2021

Contents

I Decision Trees

I.I Question 1

(a) Answer: 4. Averaging the results from multiple decision trees will decrease the variance. One good way to think about this is to make an analogy with n independent random variables Z_n each one with variance σ^2. We recall that the variance of the mean $\bar{Z}$ will then be $\frac{\sigma^2}{n}$, i.e. *averaging a set of observations reduces variance* [2]. Moreover, considering for instance, a bagging of decision trees - the usual case where we average decision trees - , each decision tree is built with a different bootstrapped training set and, thus, the final result (the average), will depend a lot less on the selection of the original training dataset, therefore reducing the variance. However, there is no particular reason why the bias should increase while averaging the decision trees. Again, using the analogy with the random variables, if $\mathbb{E}[Z] = \mu$, we can write $\mathbb{E}[\bar{Z}] = \mu$.

(b) Answer: 1 (True). Indeed, we could say that in this case with many data points with incorrect labels, Random Forests would perform better than AdaBoost. The reason is that AdaBoost's main characteristic is that it uses sequential weak classifiers that are trained on the performance of previously trained classifiers and tries to focus on difficult examples, i.e. examples that were miss classified by previous classifiers. Therefore, in this case, AdaBoost would most probably focus too much on the examples with high label noise and would, thus, learn to fit them too well. However, since the labels are incorrect, it would learn incorrectly. Since Random Forest doesn't focus so much on hard examples, it would perform better.

I.II Question 2

(a) To do the computations, we can define the following functions.

```
log2 = lambda x: log(x,2.0)
def entropy_node(prop):
    total= np.sum(prop)
    entropy_node = 0
    for i in prop:
        if i == 0:
            continue
        entropy_node -= i/total * log2(i/total)
    return total,entropy_node
def calculate_entropy(prop_1, prop_2):
    #prop_1 and prop_2 are lists. for example, 2 rejects and 0 accepts becomes [2,0]
    total_1, entropy_1 = entropy_node(prop_1)
    total_2, entropy_2 = entropy_node(prop_2)
    total = total_1 + total_2
    return total_1/total * entropy_1 + total_2/total * entropy_2
```

First, by executing $entropy\_node([6,4])$, - where $[6,4]$ is the number of Rejects and Accepts - we notice that $I(parent) = 0.9709$. Let we analyse all the possibilities of split. Then, we will consider, for example, the split $SN > 2$, to split in on side all the $SN \leq 2$ and on the other, all the $SN > 2$. Let's compute the entropy for all the possible splits of SN.

- $SN > 0$: the impurity is the $\boxed{I(parent) = 0.9709}$, since we are not actually splitting it.
- $SN > 1$: We can calculate this by using our function and calling $calculate\_entropy([2,0],[4,4])$, which will compute the entropy of each node and will do an weighted average. Here, for instance, we would get, $(-\frac{2}{2}\log(\frac{2}{2}) - \frac{0}{2}\log\frac{0}{2}) \times \frac{2}{10} + (-\frac{4}{8}\log(\frac{4}{8}) - \frac{4}{8}\log\frac{4}{8}) \times \frac{8}{10} = 0.8$. Hence, $\boxed{I(children) = 0.8}$
- $SN > 2$: The idea is the same as $SN > 1$. Here, we use $calculate\_entropy([1,4],[2,3])$ and thus $\boxed{I(children) = 0.8464}$.
- $SN > 3$: The idea is the same as $SN > 1$. Here, we use $calculate\_entropy([5,1],[1,3])$ and thus $\boxed{I(children) = 0.7145}$.
- $SN > 4$: The idea is the same as $SN > 1$. Here, we use $calculate\_entropy([6,3],[0,1])$ and thus $\boxed{I(children) = 0.8264}$.

We recall that maximizing Δ_{info} is equivalent to minimizing the impurity of the children, since $I(parent)$ is equal to all. Therefore, we choose here the split $\boxed{SN > 3}$, that has $I(children) = 0.7145$, considering the entropy.

(b) We first need to choose the best first split to make. For that, let's create a function that computes the gini index. $gini\_node(prop)$ calculates the gini for a node, by computing $1 - \sum_{i=0}^{c-1} p_i(t)^2$.

```
def gini_node(prop):
    total= np.sum(prop)
    gini = 1
    for i in prop:
        gini -= (i/total)**2
    return total,gini
```

Moreover, $calculate\_geni\_index(prop\_1, prop\_2)$ calculates the $I(children)$, by doing the weighted average of the *Gini index* of each child node.

```
def calculate_geni_index(prop_1,prop_2):
    total_1, gini_1 = gini_node(prop_1)
    total_2, gini_2 = gini_node(prop_2)
    total = total_1 + total_2
    return total_1/total * gini_1 + total_2/total * gini_2
```

Let's examine each of the possible nodes possible.

1. First split

- **CW**: For the elements with $CW = Yes$, we have 3 rejects and 2 accepts. For $CW = No$ we have the same proportion. Here, we could use $calculate\_geni\_index([3,2],[3,2])$ that would do the following computation: $(1-(\frac{2}{5})^2-(\frac{3}{5})^2)\times\frac{5}{10}+(1-(\frac{2}{5})^2-(\frac{3}{5})^2)\times\frac{5}{10}=0.48$. We have then the impurity of the children $\boxed{I(children)=0.48}$.
- **RM**: We consider the same computations as the CM split. We can thus compute $calculate\_geni\_index([2,4],[4,0])$ and we get $\boxed{I(children)=0.267}$.
- **SN**: We know from **(a)** that the best split, i.e. the one with more gain of information is the split $SN > 3$. We note here that the optimal split will not change by changing from entropy to gini, which can be easily seen by looking at the graphs of both in the page 129 of [5]. We compute here the *Gini index* to be able to compare it to the other two possible splits. We compute $calculate\_geni\_index([5,1],[1,3])$ and we get $\boxed{I(children)=0.3167}$.

We conclude then that the best split is the one with minimum $I(children)$, i.e. the **RM split**.

2. Second split

After splitting in **RM**, we get, for $RM = No$, every decision is Reject. Therefore, we will only split the side $RM = Yes$. In this node, we have 4 Accepts and 2 Rejects, which gives us $\boxed{I(parent)=0.44}$. Let's define the best split.

- **CW**: Computing $calculate\_geni\_index([1,2],[1,2])$ we have $\boxed{I(children)=0.44}$. We notice that $\Delta_{info}=0$.
- **SN**: Here, we notice that we don't have the case $SN = 1$. Therefore, we will only consider the cases $SN > 2$, $SN > 3$ and $SN > 4$
  - $SN > 2$: We compute $calculate\_geni\_index([1,1],[1,3])$ and get $\boxed{I(children)=0.417}$.
  - $SN > 3$: We compute $calculate\_geni\_index([2,1],[0,3])$ and get $\boxed{I(children)=0.222}$.
  - $SN > 4$: We compute $calculate\_geni\_index([2,3],[0,1])$ and get $\boxed{I(children)=0.400}$.

We conclude thus that the best split is in $\boldsymbol{SN > 3}$.

3. Third split

We have here 2 Rejects and 1 Accept. Thus, $I(parent) = 0.44$. Let's check what happens if we split for **CW**.

- **CW**: We compute $calculate\_geni\_index([1, 1], [0, 1])$ and get $\boxed{I(children) = 0.33}$.

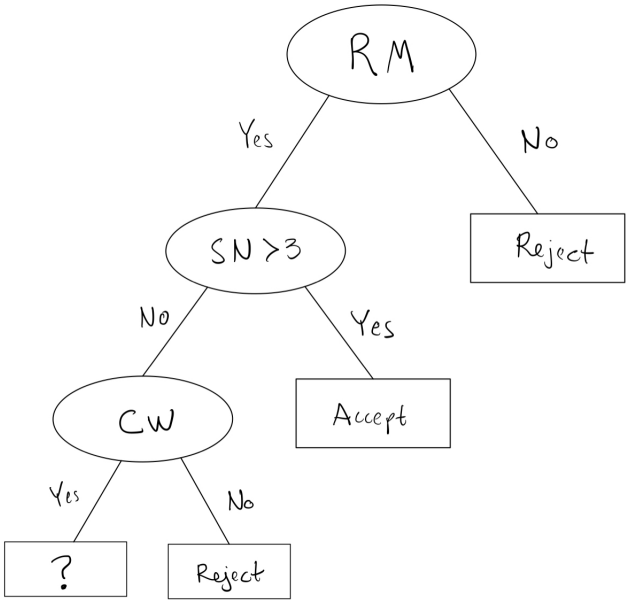

Figure 1: Decision tree developed by the algorithm

The final tree we built by the algorithm is shown above. We see that in the case $RM = Yes$, $SN \leq 3$ and $CW = Yes$, we don't know how to classify the data, since we have two data points with the same features but different labels. A solution would be to randomly choose one of them as a classification or to take the most conservative decision, which is to reject the paper. We will discuss more about this in the next item.

(c) As discussed above, one of the problems of this model is that we cannot conclude anything in one of the leafs of the tree (where $RM = Yes$, $SN \leq 3$ and $CW = Yes$), since we have two data points with the same features but different labels. This could be due to some features that are lacking in our model or maybe it is caused by some noise in our specific training data set. Again, for this particular decision tree, we could choose one of them randomly to use as a classification or choose the most conservative decision, which would be to Reject the paper. Moreover, since we built this model based on only 10 data points, it is highly probable that the model is overfitting the data and would not classify well an unseen data.

Therefore, the model is not really trustworthy and we could address these problems by:

1. Collecting more data, in order to avoid overfitting our model and to get a more general model.
2. Make some feature engineering and create some new features that could be useful in our model.

II SVMs

II.I Question 3

Answer: 3. Generally, we know that the training error tends to get lower and lower when we increase the flexibility of the model. Moreover, we know that the test error usually has a *U-shape*, since if we increase too much the flexibility the model, we will overfit the data (Figure 2) [3]. Furthermore, the training error is usually smaller than the test error, since the algorithms are optimized in order to minimize the training error. Therefore, since by increasing the degree p of the polynomial Kernel we are essentially increasing the complexity of the model, we would expect to see these characteristics in a realistic example.

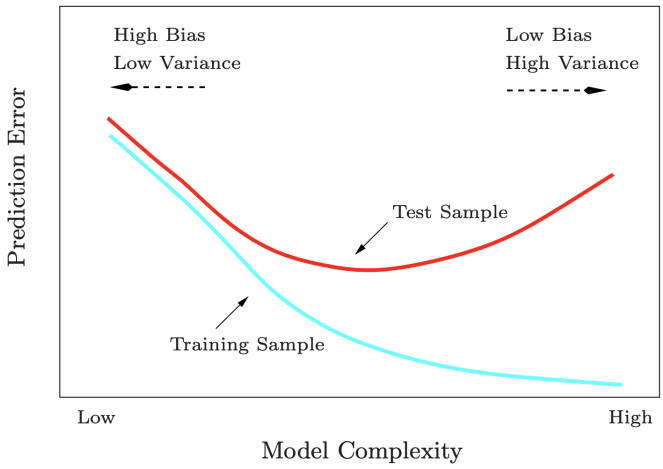

Figure 2: Prediction Error X Model Complexity

Let's thus look at each of the alternatives carefully.

1. **Not realistic.** We can see that the training error is indeed decreasing and that the test error is somewhat similar to a *U-shape* curve. However, the test error is smaller than the training error for $p = 1$, $p = 2$ and $p = 3$ which is very unlikely and, therefore, not realistic.

2. **Not realistic.** We can see that the training error is increasing with the increase of the complexity of the model, which is not realistic at all, as we can see from Figure 2. We would expect the model to fit better and better the training data as we increase its flexibility (reducing the bias) and not the opposite.

3. **Realistic.** Here, we can see all the characteristics from described above and shown in Figure 2. Therefore, it is a realistic example of what could happen while fitting the SVMs with increasing complexity.

4. **False.** As shown above, items 1 and 2 are not realistic. Therefore, the answer is False.

II.II Question 4

(a) There are two main reasons why this can be done. Firstly, since the regularization term is $\frac{C}{2}\ \sum_{i=1}^{n}\xi_i^2$, putting this constraint $\xi_i \geq 0$ would not change anything in the function we are trying to minimize. However, for the L_1 case it was strictly necessary, otherwise the optimization would always set ξ_i to $-\infty$, since it would minimize the problem, even if it is clearly not a valid solution.

Secondly, even if there is not an absolute value or a squared term in the constraint $y^i(w^Tx^i+b) \geq 1-\xi_i$, we can still drop the constraint $\xi_i \geq 0$ and this is the least trivial part. We can explain it as follows. Suppose $y^i(w^Tx^i+b) \geq 1$. Then ξ_i could potentially be negative, with the condition still being satisfied. However, it does not make sense for the optimization algorithm to do this, since we would be adding an useless positive term into a minimization problem, without any particular interest, since the condition would be satisfied any way if $\xi_i = 0$. Now, if $y^i(w^Tx^i+b) \leq 1$, the term ξ_i would need to be non-negative, since the constraint $y^i(w^Tx^i+b) \geq 1-\xi_i$ needs to be satisfied.

Another intuitive way to think about the second aspect is to think that we have a "budget" for how many ξ_i we can "spend" and that it would make no sense to "spend" them in something that would make the problem more restrictive when the constraint is already satisfied. [2]

(b) If we follow the steps explained in [1] and [4], we can use the Lagrange multipliers to express the problem as follows (Obs: we omit here the KKT conditions for simplification):

$$\boxed{\mathcal{L}(w,b,a,\xi) = \frac{1}{2}\left\|\omega\right\|^2 + \frac{C}{2}\ \sum_{i=1}^{n}\xi_i^2 - \sum_{i=1}^{n}a_i\{y^i(w^Tx^i+b)-1+\xi_i\}} \tag{1}$$

The construction is really similar to the L_1 case, where we add the term $-\sum_{i=1}^{n}a_i\{y^i(w^Tx^i+b)-1+\xi_i\}$ because we have the condition $-y^i(w^Tx^i+b)+1-\xi_i \leq 0$. Moreover, we do not have a term like $-\sum_{i=1}^{n}r_i\xi_i$ because, as explained in **(a)** , we dropped the constraint $\xi_i \geq 0$.

(c) Let's compute $\nabla_w\mathcal{L}$, $\frac{\partial\mathcal{L}}{\partial b}$ and $\nabla_{\xi_i}\mathcal{L}$ and set them to be equal 0.

$$\begin{cases} \nabla_w\mathcal{L} = 0 \implies \boxed{w = \sum_{i=1}^{n}a_iy^ix^i} \\ \frac{\partial\mathcal{L}}{\partial b} = 0 \implies \boxed{\sum_{i=1}^{n}a_iy^i = 0} \\ \nabla_{\xi_i}\mathcal{L} = 0 \implies C\xi_i - a_i = 0 \implies a_i = C\xi_i \implies \boxed{\xi_i = \frac{1}{C}a_i, \text{ if } C \neq 0} \end{cases} \tag{2}$$

(d) Now, let's compute the dual formulation of the problem, by plugging the results found in **(c)** in the equation (1). We can write then

$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{n}a_iy^i(x^i)^T\sum_{j=1}^{n}a_jy^jx^j + \frac{C}{2}\sum_{i=1}^{n}\underbrace{\xi_i^2}_{\frac{1}{C^2}a_i^2} + b\cancelto{0}{\sum_{i=1}^{n}a_iy^i} - \sum_{i=1}^{n}\sum_{j=1}^{n}a_ia_jy^iy^j(x^i)^Tx^j + \sum_{i=1}^{n}a_i - \sum_{i=1}^{n}\underbrace{a_i\xi_i}_{\frac{a_i^2}{C}} \tag{3}$$

We then get the following dual problem

$$\begin{cases} \max_a \sum_{i=1}^{n}a_i - \frac{1}{2C}\sum_{i=1}^{n}a_i^2 - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}a_ia_jy^iy^j\langle x^i, x^j\rangle \\ \text{with } \sum_{i=1}^{n}a_iy^i = 0 \text{ and } a_i \geq 0 \end{cases} \tag{4}$$

III Neural Networks

III.I Question 5

(a) We have here a perceptron that is essentially calculating $f(x) = s(w^T x)$, with $s(y)$ being -1 for $y \in (-\infty, 0]$ and 1 for $y \in (0, +\infty)$. We can see in **Algorithm 1** [5] the update proposed by Rosenblatt's training algorithm, when used in a stochastic gradient descent. We remember that x_i has a bias term included in him.

Algorithm 1 Update Rule for our case (with $s(y)$ and $\eta = 1$)

if $(s(w^T x_i) \times y_i) == 1$ **then**
  $w \leftarrow w$
else if $s(w^T x_i) == 1$ **then**
  $w \leftarrow w - x_i$
else
  $w \leftarrow w + x_i$
end if

Let's first initialize the $w^0 = (w_1^0, w_2^0, w_0^0) = (-3, 2, 1)$, where the term w_0^0 corresponds to the bias. Since only the data points (3) and (4) were miss classified during the algorithm, we know that either only these points have been selected during the algorithm or the other data points that were selected during the stochastic gradient descent were well classified and then the update $w \leftarrow w$ of **Algorithm 1** would not change anything. Therefore, we only need to analyse these two data points. Let's try to understand how the weights were updated trying to discover in which order they were miss classified.

1.
   - **Third data point**: $s(1+3-2) = +1$. We see that we miss classify this data point, since the right classification is -1. Moreover, since $(s(w^T x_i) \times y_i) \neq 1$ and $s(w^T x_i) == 1$, the update will be $w \leftarrow w - x_i$.
   - **Forth data point**: $s(1-6+4) = -1$. The data is well classified. We conclude then that the first update done was indeed the one induced by the miss classification of the third data point.

   We then do the update $w \leftarrow (-3, 2, 1) - (-1, -1, 1)$ and then $\boxed{w^1 = (-2, 3, 0)}$.

2.
   - **Third data point**: $s(0+2-3) = -1$. The data is well classified, since y_3 is indeed -1.
   - **Forth data point**: $s(0-4+6) = +1$. We see that we miss classify this data point, since the right classification is -1. Moreover, since $(s(w^T x_i) \times y_i) \neq 1$ and $s(w^T x_i) == 1$, the update will be $w \leftarrow w - x_i$.

   We then do the update $w \leftarrow (-2, 3, 0) - (2, 2, 1)$ and then $\boxed{w^2 = (-4, 1, -1)}$.

3. We see that the only point we could miss classify here is the data point (3), since we know already miss classified (3) one time and (4) one time. Indeed, $s(-1+4-1) = +1$. Again, since $(s(w^T x_i) \times y_i) \neq 1$ and $s(w^T x_i) == 1$, the update will be $w \leftarrow w - x_i$.

   We then do the update $w \leftarrow (-4, 1, -1) - (-1, -1, 1)$ and then $\boxed{w^3 = (-3, 2, -2)}$.

4. Now that we have passed throughout all the miss classifications of the algorithm, let's verify if we correctly classify all the data.

   - **First data point**: $s(-2+9+4) = +1$. We classify the data correctly.
   - **Second data point**: $s(-2+3+2) = +1$. We classify the data correctly.
   - **Third data point**: $s(-2+3-2) = -1$. We classify the data correctly.

- **Forth data point**: $s(-2-6+4) = -1$. We classify the data correctly.
- **Fifth data point**: $s(-2-3+2) = -1$. We classify the data correctly.

Therefore, we verify that after our stochastic gradient descent,**we classify all the data points correctly**.

(b) The third (-1,-1) and forth (2,2) data points would change the output of the algorithm. For example, if we removed the data point (3), our initialization w^0 would correctly classify all of our data points and we would not have a single update of the weights. As another example, if we removed the point (4), the weight w^1 would classify all of our data points correctly and we would have w^1 as our final weight. Therefore, we can conclude that, indeed, a single data point could change completely the decision boundary of our model.

(c) Adding the point $(2, -2)$ with the label $+1$ would be a huge problem, because it would mean that the data would not be linearly separable anymore, as we can see in Figure 3, where the new point is represented in the blue region and the decision boundary is the one computed in **(b)**, i.e. the one defined by w^3. As we studied in the course, a problem like this, which is analogous to the XOR problem, cannot be solved by a 1 layer perceptron. Therefore, by adding this point, we would not be able to classify correctly all our training data and we can either use this model with an error of at least one miss classification, or add more layers to the network, in order to be able to classify all our data correctly.

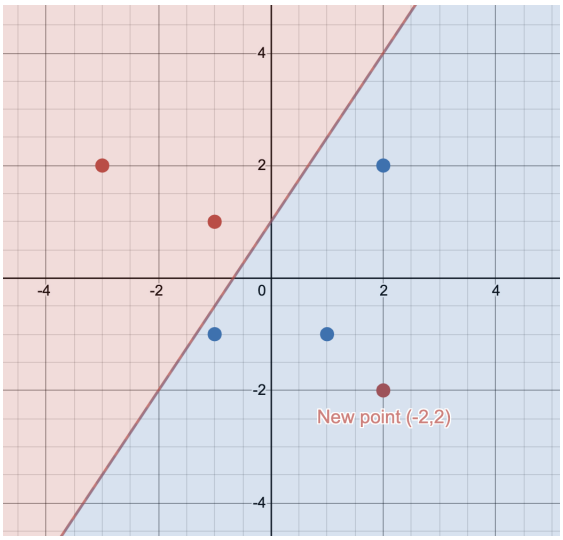

Figure 3: Representation of the points in a plane

III.II Question 6

We have a three layer fully-connected network (one input layer L_1, one hidden layer L_2 and one output layer L_3) with the sigmoid as activation function, i.e. $\sigma(x) = \frac{1}{1+e^{-x}}$. Moreover, the loss function is the MSE , i.e. $E = \frac{1}{n_3}\sum_{i=1}^{n}(t^j - y_3^j)^2$, where t^j is the true label and y_3^j is the j^{th} element of the output, i.e. our prediction.

(a) We need to find $\frac{\partial E}{\partial z_3^j}$ in terms of y_3^j and t^j. We can do this by using the chain rule and by noticing that $y_3^j = \sigma(z_3^j)$ and that $\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$. We get then:

$$\frac{\partial E}{\partial z_3^j} = \frac{\partial E}{\partial y_3^j}\frac{\partial y_3^j}{\partial z_3^j} = -\frac{2}{n_3}(t^j - y_3^j)\, y_3^j\,(1-y_3^j) = \frac{2}{n_3}(y_3^j - t^j)\, y_3^j\,(1-y_3^j) \tag{5}$$

Therefore, $\boxed{\frac{\partial E}{\partial z_3^j} = \frac{2}{n_3}(y_3^j - t^j)\, y_3^j\,(1-y_3^j)}$

(b) We need to find $\frac{\partial E}{\partial y_2^k}$ in terms of $\frac{\partial E}{\partial z_3^j}$ and W_{23}. Since $z_3^j = \sum_{k=1}^{n_2} y_2^k\, W_{23}^{kj}$. Then, for a fixed k:

$$\frac{\partial E}{\partial y_2^k} = \frac{\partial E}{\partial z_3^j}\,\frac{\partial z_3^j}{\partial y_2^k} = \frac{\partial E}{\partial z_3^j}\, W_{23}^{kj} \tag{6}$$

Therefore, $\boxed{\frac{\partial E}{\partial y_2^k} = \frac{\partial E}{\partial z_3^j}\, W_{23}^{kj}}$

(c) We need to find $\frac{\partial E}{\partial W_{23}^{kj}}$ in terms of y_2^k, y_3^j and t^j. By using the chain rule again we have:

$$\frac{\partial E}{\partial W_{23}^{kj}} = \frac{\partial E}{\partial y_3^j}\,\frac{\partial y_3^j}{\partial z_3^j}\,\frac{\partial z_3^j}{\partial W_{23}^{kj}} = \frac{2}{n_3}(y_3^j - t^j)\, y_3^j\,(1-y_3^j)\, y_2^k \tag{7}$$

Therefore, $\boxed{\frac{\partial E}{\partial W_{23}^{kj}} = \frac{2}{n_3}(y_3^j - t^j)\, y_3^j\,(1-y_3^j)\, y_2^k}$

IV Unsupervised Learning

IV.I Question 7

(a) Answer: 1 and 2. This question asks us about the effects of using the Principal Component Regression (PCR) technique, that consists of executing a PCA and then apply a regression technique. Let's examine each of the alternatives carefully.

1. **True.** This technique normally leads to a less noisy result, since we often have the signal (as opposed to the noise), concentrated on the first few principal components. [2] Therefore, we reduce the dimensions, considering a linear combination of the variables in order to maximize the variance (and therefore the signal captured), avoiding fitting our model in some noise, which could lead to overfitting.

2. **True.** Since we reduce the dimension from p, for example, to M, with $M << p$, we have a lot less variables to consider, which reduces the running time.

3. **False.** PCA does not handle missing data. Furthermore, it is not even capable of doing the computations (at least in *scikit-learn*) if you input data with missing values. Therefore, you need to handle missing data separately (by using some kind of imputer, for example), as it would be needed in a normal regression model.

4. **False.** PCA only computes a linear combination of the original features in order to maximize the variance. Therefore, it is not capable of transforming a non-linear data (for example, a circle) in a linearly separable data.

(b) Answer: 2 and 4. Let's examine each of the alternatives separately, remembering that $X = U\Sigma V^T$, where U and V are orthogonal and Σ is diagonal with the singular values.

1. **False.** Firstly, note that the matrix X^TX is symmetric, since $(X^TX)^T = X^TX$. Moreover,

$$X^TX = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma^T \underbrace{U^TU}_{=\mathbb{I}} \Sigma V^T = V\Sigma^T\Sigma V^T \tag{8}$$

   Therefore, we get the Eigendecomposition of the matrix X^TX, where the singular values squared of the matrix X are the eigenvalues and the columns of the matrix V are the eigenvectors. Therefore, since the questions asks about the matrix U, this is false.

2. **True.** We will follow the same idea of the first item. We note again that XX^T is symmetric, since $(XX^T)^T = XX^T$ and that, thus, the eigendecomposition exists. Then

$$XX^T = (U\Sigma V^T) (U\Sigma V^T)^T = U\Sigma \underbrace{V^TV}_{=\mathbb{I}} \Sigma^T U^T = U\Sigma\Sigma^T U^T \tag{9}$$

   We again have the Eigendecomposition of the initial matrix (XX^T). Here, the eigenvectors of XX^T are indeed the columns of U. Therefore, it is True.

3. **False.** We note again that X^TXX^TX is symmetric, since $(X^TXX^TX)^T = (X^TX)^T(X^TX) = X^TXX^TX$. Then, using the result from the item 1, we have

$$X^TXX^TX = V\Sigma^T\Sigma \underbrace{V^TV}_{=\mathbb{I}} \Sigma^T\Sigma V^T = V\Sigma^T\Sigma\Sigma^T\Sigma V^T \tag{10}$$

   Again, we get the Eigendecomposition, where the eigenvalues of X^TXX^TX are σ_i^4 (where σ_i are the singular values of X) and the eigenvectors are the columns of V (not U). Thus, it is False.

4. **True.** We note again that XX^TXX^T is symmetric, since $(XX^TXX^T)^T = (XX^T)^TXX^T = XX^TXX^T$. Then, using the result from item 2, we have

$$XX^TXX^T = U\Sigma\Sigma^T\underbrace{U^TU}_{=\mathbb{I}}\Sigma\Sigma^TU^T = U\Sigma\Sigma^T\Sigma\Sigma^TU^T \tag{11}$$

Again, we get the Eigendecomposition, where the eigenvalues of XX^TXX^T are σ_i^4 (where σ_i are the singular values of X) and the eigenvectors are the columns of U. Thus, it is True.

IV.II Question 8

(a) In this example, we directly realize that all the three points, $(-1,-1),(0,0)$ and $(1,1)$ are points of a line described by the equation $y = x$. Therefore, we know that all the variability of the data can be explained only by one dimension, as we will see in the next items.

Therefore, we can conclude that the vector defining the line is $v = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$, since the vector needs to be normalized (otherwise we would be able to increase the variance arbitrarily by choosing a vector with elements arbitrarily large)[2]. Thus, the first principal axis is $\boxed{v = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})}$

(b) Firstly, we note that the data is already centered. Then , we can find the coordinates of the data points in one dimension by projecting the points in our one dimensional space defined by $v = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$. Since v is already normalized, we can project the points only by doing an inner product. We have then

$$\begin{cases} (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}) \cdot (-1,-1) = -\sqrt{2} \\ (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}) \cdot (0,0) = 0 \\ (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}) \cdot (1,1) = \sqrt{2} \end{cases} \tag{12}$$

We are going to call this components z_{i1}, where 1 tells us that it belongs to the first principal component and i references each datapoint. Therefore, $\boxed{z_{11} = -\sqrt{2}}$, $\boxed{z_{21} = 0}$ and $\boxed{z_{31} = \sqrt{2}}$

(c) Since the data is centered, we can do

$$Var_{proj} = \frac{1}{n}\sum_{i=1}^{n} z_{i1}^2 = \frac{2+2}{3} = \frac{4}{3} \tag{13}$$

We can also verify our early conclusion that the projected data would capture all the variance of the data by computing

$$\sum_{j=1}^{2} \frac{1}{3} \sum_{i=1}^{3} x_{ij}^2 = \frac{1}{3}(1+0+1) + \frac{1}{3}(1+0+1) = \frac{4}{3} \tag{14}$$

Therefore, we see that both variances are equal and, in particular, $\boxed{\text{Var}_{proj} = \frac{4}{3}}$

(d) Since our projected data in 1-D explains all of the variance of the data (cf item **(c)**), we can imagine that the error will be zero. Let's prove it. Let $\mathcal{A} = \{(0,1),(1,0)\}$ and $\mathcal{B} = \{(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}), (-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})\}$ be two basis on $\mathbb{R}^2$. $\mathcal{B}$ represents the basis of our PCA (essentially a 45^{o} rotation of $\mathcal{A}$) and $\mathcal{A}$ represent the original base (canonical). Note that our 1 dimensional space , defined above, is given by the first principal component, which is the first element of $\mathcal{B}$ and, thus, the point z_{11}, for example, will be represented here as $[\sqrt{2}, 0]^T$, since we only care about the first component (even if we know the second components would all be equal to 0 anyway, since the data if fully explained by 1 dimension). We can define the change of basis matrix $S_{\mathcal{B}\to\mathcal{A}}$ as

$$S_{\mathcal{B}\to\mathcal{A}} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \tag{15}$$

And then, applying it to our "one-dimensional" projected data $z_{11} = [-\sqrt{2}, 0]^T$, $z_{21} = [0,0]^T$ and $z_{31} = [\sqrt{2}, 0]^T$, we get $\boxed{S_{\mathcal{B}\to\mathcal{A}}\, z_{11} = (-1,-1)}$, $\boxed{S_{\mathcal{B}\to\mathcal{A}}\, z_{21} = (0,0)}$ and $\boxed{S_{\mathcal{B}\to\mathcal{A}}\, z_{31} = (1,1)}$, i.e. we recover exactly the original points with $\boxed{\text{Error} = 0}$.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[2] T. Hastie G. James, D. Witten and R. Tibshirani. *An Introduction to Statistical Learning with applications in R*. Springer, 2017.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[4] Andrew Ng. *CS229 Lecture notes*. Stanford.

[5] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, us ed edition, May 2005.