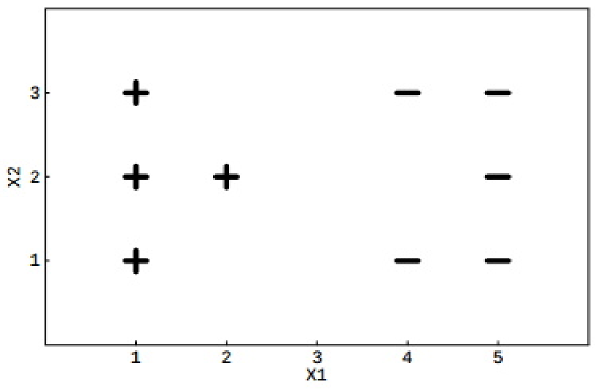

Model final exam

$\boxed{1}$ (*25 bodov*) **Short questions**

Answer the following questions and **briefly justify your answers.**

a) In which combinations it is advisable to use bagging or boosting and why?
☐ bagging together with shallow decision trees
☐ bagging together with deep decision trees
☐ bagging together with SVM with Gaussian kernel
☐ boosting together with shallow decision trees
☐ boosting together with deep decision trees
☐ boosting together with logistic regression

b) In the following figure showing positive and negative training examples, draw the classification boundary obtained using a simple SVM with linear kernel. Circle all data points for which, if removed from the training set, the classification boundary would change. Briefly justify.



c) It is common to use "bag of words" to represent the text as a feature vector. Each feature corresponds to a single word from a dictionary and its value is 1, if the word occurs in the text, and 0 otherwise. Prof. Knowitall says that with SVMs, we get the same effect by using kernel function $K(x, y) = \#(x \cap y)$, where $\#(x \cap y)$ is the number of words that are common in both x and y.

Prof. Knowitall is ☐ correct ☐ incorrect

d) In generalized linear regression, which factor affects the tradeoff between underfitting and overfitting the most? ☐ using or not using the feature with a constant value

☐ degree of the polynomial used to expand the original feature vector

☐ choice of the training algorithm (gradient descent vs. solving a system of linear equations)

☐ amount of Gaussian noise present in the data

e) Consider a hypothesis class H, defined over R^2 space, where each hypothesis is a square with axis aligned boundaries (points inside the square and on its boundary are classified as positive).

VC dimension of class H is: ☐ < 4 ☐ $= 4$ ☐ > 4

$\boxed{2}$ (*25 bodov*) **Card game**

In a card game, in every move we draw a card with value 0, 2, or 3, each with probability 1/3. Before each move, you can say that you are finished (F) or that you will continue playing (C). If sum of the cards that you draw will surpass value of 6, you lose and you get 0 points. If you decide to finish with the sum of less than 6, you will get the number of points equal to **three times the sum** of the cards.

We can represent this game with a Markov decision process, with the states corresponding to the sum of the cards drawn so far, i.e. $\{0, 2, 3, 4, 5, 6, \text{too much}\}$ (when the state "too much" is reached, we finish the game automatically).

a) Consider the following policy:
$\pi(0) = \pi(3) = \pi(5) = C$
$\pi(2) = \pi(4) = \pi(6) = F$
Write the equations that characterize the value function V^π for this policy.

$$V^\pi(0) =$$
$$V^\pi(2) =$$
$$V^\pi(3) =$$
$$V^\pi(4) =$$
$$V^\pi(5) =$$
$$V^\pi(6) =$$
$$V^\pi(\text{too much}) = 0$$

b) Solve the equations from task a)

	0	2	3	4	5	6	too much
V^π							

c) Write Bellman equation characterizing optimal value function V^*:

$$V^*(0) =$$
$$V^*(2) =$$
$$V^*(3) =$$
$$V^*(4) =$$
$$V^*(5) =$$
$$V^*(6) =$$
$$V^*(\text{too much}) = 0$$

d) Use the value iteration with synchronous update (all the values are updated at the same time) to compute the value function. For simplicity, **round all the intermediate results to the nearest integer.**

iterácia	0	2	3	4	5	6	too much
1	0	6	9	12	15	18	0
2							
3							
4							
5							
6							

e) What is the optimal policy and what is the expected number of points that you get from the game?

$\boxed{3}$ (*25 bodov*) **What is this?**

In a co-worker's code, you have dicovered a section which takes on the input $t \times n$ array `train`, a vector `ytrain` with t values and $m \times n$ array `test`. Array `train` represents t training examples, each with n features; for each training example, there is a corresponding target value in the array `ytrain`.

The code looks as follows:

```
import numpy as np
t=train.shape[0]
np.append(train,np.ones((t,1)),axis=1) # ****
np.append(test,np.ones((t,1)),axis=1)  # ****
n=train.shape[1]
w=np.linalg.inv(train.T.dot(train)).dot(train.T).dot(ytrain)
result=test.dot(w)
```

a) Which algorithm from the class does this algorithm reminds you of? What is the result of the code?

b) What are the lines denoted by **** good for?

c) Prof. Knowitall says that line starting with w should be replaced with an alternative which looks as follows:

```
w=np.zeros(n)
for i in range(100):
  error=train.dot(w)-ytrain
  for j in range(n):
      delta=1/t*......
      w[j]=w[j]-0.1*delta
```

Explain this code and fill in the missing part.

d) When would it be better to use the original code and when it is better to use Prof. Knowitall's code? Why?

$\boxed{4}$ (*25 bodov*) **Stock price prediction**

Assume that you have the data about stock price development over time. For each stock, you have data about its price at close for every day for the time period of at least one year.

Your task is to propose a system that predicts whether a particular stock should be bought or sold on a particular day.

a) Formulate the problem as a standard classification problem with feature vectors of fixed length. What features would you use and why?

b) How would you evaluate the performance of your system? How would you split the data into the training and testing set? Can you relate your proposal to basic concepts of machine learning theory?

c) It is always necessary to compare the performance of a newly proposed system to a simple baseline strategy. What would you propose as the baseline strategy and why?

d) How would you diagnose whether your system suffers from overfitting or underfitting?

e) There are global events affecting stock exchange as a whole (i.e. black days, bubble bursts, international trade war declarations, ...). Can your proposed system take these events into account? If not, can you propose a modification?

f) What other data would be useful to incorporate in such system? How would you get such data and how would you incorporate them?