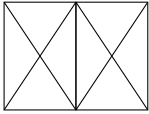


Exam for the lecture
„Machine Learning in Finance“
401-3932-19S

Please fill in the following details:

Surname	Given name		Legi-number	Prüfungsnr.
<input type="text"/>	<input type="text"/>		<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>		<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>		<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>		<input type="text"/>	<input type="text"/>

the first two letters for each box *last six numbers* *Do not fill this out*

Important:

- Fill in the first two letters of your surname, your given name and the last six numbers of your legi-number
- Put your student card on the table
- The duration of the exam is **90 minutes**. Before these 90 minutes start, you are allowed to read the exam for **10 minutes** during which no writing is allowed. Follow the commands indicating whether you can start reading and start writing. When you are told to **stop** writing, put down your pens immediately.
- Begin each exercise on a new sheet of paper, and write your name on each sheet. There are 4 questions in total.
- Only pen (ink) and paper are allowed. The ink must not be red or green. Do not use whiteout, instead just cross out the relevant parts.

Please do not fill in the following table

Question	Points	Control	Maximum
#1			
#2			
#3			
#4			
Total			

Good luck!!!

Please turn the page!

Some hints:

1. Recall that we denote with $\mathcal{C}(\mathcal{X})$ the set of all continuous functions $f : \mathcal{X} \rightarrow \mathbb{R}$. If $\mathcal{X} = \{x_1, \dots, x_n\}$ is some finite set, then we can identify $\mathcal{C}(\mathcal{X})$ with \mathbb{R}^n .
2. In Question 2-4, you will have to write pseudo code. This means that you are supposed to describe in a structured way how your algorithm works. It is not necessary to have precise syntax. Hence, you can also use mathematical notation if it is clear how it is meant (e.g. subscripts, powers and standard functions like $\sqrt{\cdot}$ etc). One example would be that if you have a vector $\alpha \in \mathbb{R}^n$, you write the sum of the elements either by (you can use Greek letters etc)

$$\text{value_sum} = \sum_{i_1}^n \alpha_i$$

or you could write

```
value_sum = 0
for i = 1, ..., len(alpha):
    value_sum = value_sum + alpha_i
```

where these are just examples. Another example of a function you may use is the function `cumsum` that computes the cumulative sum of a vector $v \in \mathbb{R}^k$, i.e.

$$\text{cumsum}(v) = w \in \mathbb{R}^k \quad \text{with} \quad w_i = \sum_{j=1}^i v_j.$$

You may further use any `python` (including `numpy`), `Matlab` or `R` syntax, i.e. in order to define a vector $\tau = (0, 1, \dots, n)$, you can e.g. write either of the following without further commenting,

- `tau = linspace(0, n, n+1)`
- `tau = arange(n+1)`
- `tau = (0, 1, ..., n)`

Use a `%` to write comments, e.g. if you explain what a function you use does. You can structure your code by defining functions, i.e. the following is admissible:

```
% randomnormal(k) returns a random vector of
% length k that is standard normal distributed
v = randomnormal(10)
solution1, solution2 = foo(v)

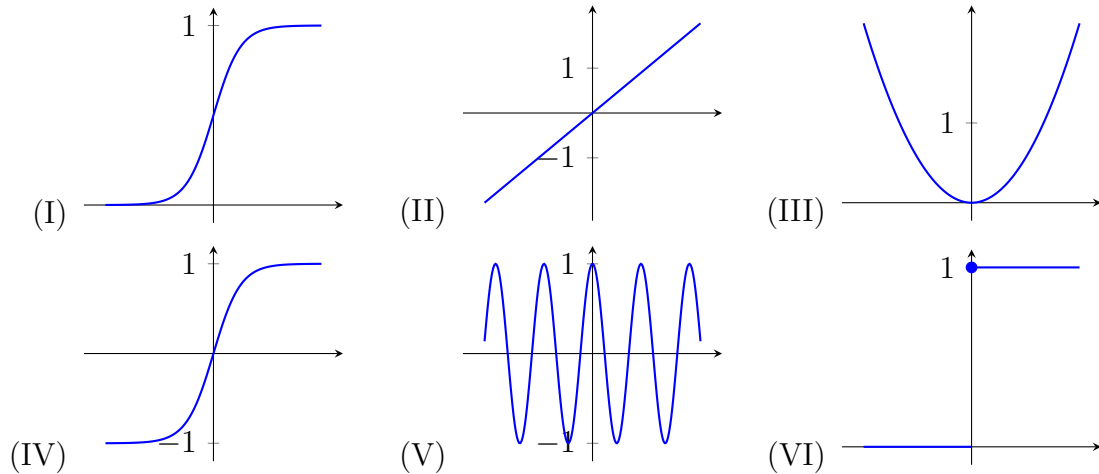
function foo(x):
    % sin and cos for a vector x is applied
    % component wise
    return (sin(x), cos(x) )
```

which generates a standard normal random vector $v \in \mathbb{R}^{10}$, and returns two vectors with components being the sin (stored in `solution1`) and cos (stored in `solution2`) of the corresponding components of the random vector. The goal of your pseudo code is only to show how your algorithm works, so do not spend time on thinking about whether this syntax is correct in some programming language.

Question 1.

(13 Points)

- (a) Give the precise definition of a discriminatory function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$. Which of the following graphs of functions and their likely extensions to $\pm\infty$ are correct examples of sigmoidal activation functions as defined in the lecture? Which of these functions are discriminatory? Justify your answers.



[5 Points]

- (b) Let $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ be a sigmoidal activation function. Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be a shallow neural network. Compute all partial derivatives of f wrt all parameters of the neural network and then wrt the input variable $x \in \mathbb{R}^d$. You can assume for this part that σ is smooth.

[4 Points]

- (c) Why are shallow neural networks of interest in machine learning: explain the universal approximation theorem. What are deep neural networks? How can we calculate derivatives of deep neural networks with respect to parameters and inputs: include the role of the backpropagation algorithm in this context.

[4 Points]

Hint: These answers do not need to be technical. You may assume that neural networks are mappings from the unit cube to the real numbers ($X = [0, 1]^d$) for some $d \in \mathbb{N}$.

Question 2.

(14 Points)

- (a) Deep portfolio optimization (without transaction costs): what is a financial market in discrete time? What is a self-financing trading strategy with initial capital x ? Describe the value function of a portfolio, and describe the expected utility of a portfolio. Formulate the optimization problem without transaction costs in generality. How does machine learning help to solve the problem approximately?

[4 Points]

For the rest of this question, you can assume the single stock case, i.e. at each time, all of the wealth is distributed between a single risky stock S and a risk-free bank account B , which is assumed to be constant one, $B \equiv 1$.

- (b) What does it mean if the underlying price process happens to be a martingale? Does it make sense to invest in such a market? Elaborate!

[2 Points]

- (c) What does it mean (does not need to be technical) that the underlying price process S is (discrete-time) Markovian? Under such a Markovian assumption, how can we simplify our parametrization of the machine learning problem?

[2 Points]

- (d) Consider an N step model (S_n) of conditional binomial form, i.e. the parameters of the model dependent on a fixed two state Markov chain (X_n) ,

$$P[S_{n+1} = S_n u | S_n, X_n] = p(X_n); P[S_{n+1} = S_n d | S_n, X_n] = 1 - p(X_n),$$

where $u > 1 > d > 0$ and an exponential utility function $U(x) = 1 - \exp(-x)$.

Write down in pseudocode an algorithm that learns the optimal portfolio with initial capital x for $T = N$ trading days using M generated trajectories of the binomial market model. For the training part, use stochastic gradient descent with step size $\gamma > 0$ and make $K \in \mathbb{N}$ epochs. Gradients are computed with mini-batches of size one.

[6 Points]

Hint: You may use a function `phi(x, theta)` that implements a smooth neural network with input $\mathbf{x} \in \mathbb{R}^n$ where you can specify the input dimension n before using the function and $\theta \in \mathbb{R}^L$ is a parameter vector corresponding to the weights of the neural network. You do not need to specify an architecture or say what L is, you can assume `phi` is suitable for approximating any continuous function. If you need multiple neural networks, write them as `phi_i` for $i \in \mathbb{N}$. In that case, specify each input dimension with n_i , the corresponding weights with θ_i .

The trajectories are stored in a Matrix $\mathbf{S} \in \mathbb{R}^{M \times (N+1)}$.

Finally, you can use derivatives such as gradients at will. E.g. if you have a value `cost`, that depends in some way on the values of θ , then `grad = ∇θ cost` computes the gradient w.r.t. the weights θ and stores the gradient in `grad` for the current values of θ . If the values of θ change, you may assume that the value of `grad` changes automatically to the gradient evaluated at the new value of θ .

Question 3.

(18 Points)

- (a) Calibration of model: describe the calibration problem as inverse problem of selecting a model given some data and some pool of models.
[2 Points]
- (b) Exemplify the ambitious and modest approach in case of the Heston model: parametrize the Heston model and describe in detail which maps can be learned and how you would do it.
[8 Points]
- (c) What is a local volatility model and what are they for: explain Dupire's formula and its shortcomings.
[2 Points]
- (d) Write down pseudocode to learn local volatility for finitely many given prices: parametrize local volatility by a neural network and solve the pricing equation by an Euler scheme, then define a loss function and write down the optimization problem that one needs to solve such that model prices and market prices are close. Discuss mini-batching in this approach, does it work?
[6 Points]

Question 4.

(16 Points)

- (a) Formulate an infinite horizon, time-homogeneous Markov decision problem (MDP) with a finite number of states, discounting factor γ and reward function R , in abstract terms. State (DPP), the HJB equation, and explain value iteration, policy iteration and Q learning.

[8 Points]

Consider the following real world problem, which is called the pricing of a *passport option*: consider one underlying S and a bank account process $B = 1$ and an investor who can go long or short by one position in the underlying in a self-financing way: goal is to optimize the positive part of the outcome

$$x_T = x_0 + (\pi \bullet S)_T = x_0 + \sum_{t=0}^{T-1} \pi_t (S_{t+1} - S_t)$$

at maturity time T , with strategy $\pi_t \in \{-1, +1\}$.

Assume that the price process follows a finite state, discrete time Markov process S with states y_1, \dots, y_n and denote by $z_1, \dots, z_{\bar{n}}$ the set of possible states of x .

- (b) Consider $\{0, \dots, T\} \times \{y_1, \dots, y_n\} \times \{z_1, \dots, z_{\bar{n}}\} \cup \{\infty\}$ as state space, a time-homogeneous Markov process which models the price process S together with time and x and which ends in ∞ after T steps. Define additionally a reward function $R(T, x, s) = R(T, x) := \max(x, 0)$, $R(t, x) = R(\infty) = 0$ for $t < T$. How does the associated MDP correspond to the valuation problem of a passport option?

[4 Points]

- (c) Consider a policy iteration algorithm for this problem: how can you represent the policy? Write down pseudo code to learn the strategy via policy iteration.

[4 Points]