Exercise 1: Reviewing High School Knowledge: Analytical Optimization

Summer Term 2024

This exercise reviews some very fundamental optimization issues, which you have already studied in school and during your first semesters.

Review: Review the terminology that we have discussed in the class:

1. Please, provide two different definitions of a maximum (and minimum, respectively):

2. What is the difference between a minimization and a maximization task?

To Do: Please, locate the optima of the following three one-dimensional functions. In so doing, discuss whether your solutions are minima or maxima. What are the particular characteristics and/or problems of the four functions?

Tasks: Please consider the following problems:

1. $f(x) = 5x^2 - 2x + 10$
2. $g(x) = x^3 - 6x^2 + 12x - 7$
3. $h(x) = (x^2 - 8x + 16)(x - 1)$
4. $i(x) = \sin(x)$

Have fun, Theo and Ralf.

Exercise 2: From Simple Texts to Analytical Descriptions

Summer Term 2024

This exercise should help you derive an analytical problem formulation from a plain and potentially vague text description.

Review: What are the necessary "ingredients" of an optimization task?

1.
2.
3.
4.

To Do: For every single task, you should discuss the following questions:

1. What is the optimization goal?
2. What are the parameters x_i to be optimized?
3. What is a suitable fitness formulation?
4. Does only one good fitness formulation exist or do you have several choices?

Applications: Please consider the following applications:

1. Assume that you have a ball-throwing catapult. The throwing speed v is constant and cannot be changed. How can you hit a target at distance d? Then, assume that you can adapt the throwing speed v. How can hit the ball the target at distance d with the minimal speed v possible?
2. A *cylindric* bucket with one bottom and no top is needed that has a pre-specified volume. The goal is to determine both the bucket's diameter and its height such that the bucket's surface is minimal.
3. You have a fence of 500 meters in length with which you are supposed to surround a rectangular area as large as possible.

Have fun, Theo and Ralf.

Exercise 3: Real-World Applications

Summer Term 2024

In the previous exercises, you have reviewed some basic skills on optimization. Now, this exercise is devoted to the understanding of how to deal with *real-world* applications. A key point is that often, the fitness $f(x_1, \ldots, x_n)$ is not directly a function of its parameters x_i, but rather in some indirect way. This fact often requires some *indirect thinking* of how to formalize a proper fitness function, and in turn, requires *practice*.

Review: Again, review the requirements/skill that are essential for almost any optimization task.

To Do: In the following applications, the focus is on the identification of the parameters x_i to be optimized, the identification of the optimization goal, the definition of a proper fitness function, and an experimental setup for the actual fitness evaluation. Please note that this exercise does not ask for the actual optimization process or choosing a particular algorithm.

Applications: Please consider the following applications:

1. A color sample is given (you can look at it) for which you do not know the red, green, and blue values. Some sort of machinery allows you to generate arbitrary colors, which you can compare with the sample. How can you determine the color components?

2. You should design the *shape* of a water pipe, which deviates the water flow by 90°. The pipe's cross-section is of circular shape with a fixed diameter. The pipe's friction should be minimal.

3. The famous Formula-1 racer Sterling Moos wanted to win the Monte Carlo race, and asked his mechanic to prepare the six gears, with each having two cogwheels, of his gearbox. How would you proceed?

4. In an experimental laboratory test, the following measurements have been obtained:

x	-2.0	-1.0	0.0	1.0	2.0	3.0	4.0	5.0	6.0
y	12.1	4.9	0.2	-3.0	-4.1	-3.2	0.1	5.1	11.9

   For further processing, the laboratory staff wants to have a quadratic function $f(x) = ax^2 + bx + c$ that describes the data as best as possible.

5. A new synthetic material has been developed, which requires a production time of 30 minutes at varying temperatures. It is known that the material's performance significantly depends on the particular temperature schedule within those 30 minutes, and thus the production line allows for resetting the temperature minute after minute.

Have fun, Theo and Ralf.

Exercise 4: Semi-Analytical Optimization Tasks

Summer Term 2024

At first glance, the optimization problems of this exercise look like analytical problems, **but** ...

Review: So far, we have considered two different types of optimization problems. These were:

1.

2.

To Do: Please, try to find the optima of the following applications, which are all given in an analytical form. Please answer the following questions for every task.

1. What are the difficulties?
2. How would you try to proceed?

Applications: Please consider the following problems:

1. $f(x) = x^2 + \sin(x)$
2. The figure presented below depicts an electrical power supply network consisting of a transformer (T), four houses ($H^{1..4}$), and three distribution nodes ($D^{1..3}$). The positions, i.e., the x and y coordinates, of the transformer and the houses are fixed and can be found in the figure. The positions of the distribution nodes, however, are flexible. The goal is to optimize the network such that its total length is minimal.

   According to Pythagoras, the distance l_{pq} between two points p and q is given as: $l_{pq} = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$. What is the total length L of the entire network? Please, write down its formula: $L(D^1_x, D^1_y, D^2_x, D^2_y, D^3_x, D^3_y)$ of the entire network.

   Questions: Can you solve the equation for its six parameters? What is the problem?

Have fun, Theo and Ralf.

Exercise 5: Global Search

Summer Term 2024

In this exercise, we study a very simple global optimization procedure, known as *systematic* or *exhaustive* search.

Review: Assume, you have a fitness function $f(x)$ of any kind, for *example* $f(x) = 0.5x^2$ or $f(x) = \frac{(x-2)(x-4)}{(x+5)(x^2-12)} + xe^{-x}$, and a search space $x_{\min} \leq x \leq x_{\max}$. How many test, i.e., function evaluations, have you do to in order to approach the location of the true optimum x_{opt} with a precision $\epsilon \leq |x_{\text{opt}} - x^o|$? How does the number of function evaluations required to reach a certain precision ϵ depend on the actual fitness function $f(x)$?

To Do: An in-depth analysis of the systematic search procedure.

Tasks: Please do the following tasks.

1. Now, let us consider a simple test case: (1) the number of test points in one dimension is $(x_{\max} - x_{\min})/\epsilon = 1000$, (2) you have a quite fast processor operating at 10 GHz, and (3) the implementation is extremely efficient and requires only ten instructions per fitness evaluations. Please, answer the following questions:
   (a) How many fitness evaluations can the processor perform per second?
   (b) **Guess** (just answer what you have in mind), in how many dimensions n, i.e., $f(x_1, \ldots, x_n)$, can this system complete its optimize task within two days? How many dimensions n can be solved and two years?
2. We now do the same task a bit more systematic. In order to go beyond "just guessing", you might be completing the following table:

n	1	2	3	4	5	6	7	8	9	10	20	50	100
time													

3. What does the results mean and imply? Please, discuss the following questions:
   (a) Now you know it: in how many dimensions n can this system complete its optimize task within two days and two years, respectively?
   (b) How well does this procedure scale?
   (c) What is the scaling law? $t(n) = O(\quad)$
   (d) What is the procedure's utility when considering real-world applications?

Have fun, Theo and Ralf.

Exercise 6: Optimization Based on Human Intelligence and Intuition

Summer Term 2024

In this exercise, you will be competing against all other class mates. To this end, you will be using a little program. After starting it, it reads two integer numbers (e.g., "`1 2`" but not "`4, 5`") per input line, and reports the corresponding fitness value, also called function or objective value. Your task is to find the location of the minimum, which has the fitness value $f(x_{\min}, y_{\min}) = 0$.

Review: In the class, we have discussed the basic principles of optimization. Please, answer the following questions:

1. How will you be deriving the next test point from the knowledge you have gathered in the past?
2. How will you be handling two dimensions?
3. How do you want to determine you step size?

To Do: Locate the minimum of each of the following two test functions. Please note that the x and y coordinates of the minimum are within the interval $[-500, 1000]$.

Tasks: For the fastest one, we offer a coffee ☕.

1. Start the program . . . . . and locate the minimum as quickly as possible.
2. Now, start the program . . . . . and search again for the minimum.

x	y	$f(x,y)$

x	y	$f(x,y)$

Have fun, Theo and Ralf.

Exercise 7: Implementation of a Simple Evolution Strategy

Summer Term 2024

It is now time to do your own programming stuff. The goal is to have a running evolutionary algorithm with which you can do further exercises. In order to ease this task quite a bit and in order to focus on the algorithmic issues rather than on tedious implementation details, we have prepared a framework for you. Since this exercise still requires some time, you are expected to do the programming as a **homework** and to bring the result to the next class!

Review: In this class, we distinguish between three different types of optimization problems:

1.
2.
3.

What type is in the focus of this class? Why not the others?

To Do: Implement a running evolutionary algorithm step by step.

Tasks: Happy programming!

1. Download the prepared material from `http://www.imd.e-technik.uni-rostock.de/ma/rs/lv/hosc/ea.zip`.
2. **Read the API documentation.** You are strongly advised to do so!
3. Implement the `main()`-function of the simple `ea`-program. The fitness function should be simply the sum of the squared x_i, which is also known as the sphere model. Where does this function has its minimum? Test the program by running it for a few generations in order to see whether you did everything correctly or not. As a recommendation, try $\mu = 1$, $\lambda = 6$, and $n = 5$ dimensions. Now, try also other values for μ, λ, and n in order to test your program a bit more.
4. To get a deeper understanding of the nature of an evolutionary algorithm, you should now implement the `mutation_offspring()`-routine by yourself. Also, implement a simple uniform crossover operator that may work on two successive offspring.
5. You are now almost ready to solve the minimization of the power distribution network from Exercise 4. Replace the fitness function (i.e., the sphere model as noted above), by the calculation of the total network length (as described in Exercise 4). Questions:
   (a) How many parameters x_i do you need?
   (b) How can you map those parameters onto the specification of the distributors D_i?

   Test your program by doing some runs and report your minimal network length to the research assistant.

Have fun, Theo and Ralf.

Exercise 8: Performance of the Simple Evolution Strategy

Summer Term 2024

This exercise explores to what extent the performance of an evolutionary algorithm depends on its basic parameters.

Review: What are the basic parameters that *configure* an evolution strategy?

1.

2.

3.

Please, recall how the mutation operator distributes offspring around the parent.

To Do: Basically, this exercise requires two different types of work. First, you have to adapt your program, and then you have to run some experiments. You may complete the table printed below.

Tasks:

1. Recycle the first version of your personal implementation of the simple evolution strategy. This (test) version was using the sphere model $f(x_1, \ldots, x_n) = \sum_i x_i^2$ as the fitness function.

2. *Theoretical work:* Assume, for the moment, that you have just one parent, i.e., $\mu = 1$. Now, discuss how an increasing number λ of offspring changes their distribution around the parent. How does this affect the rate of progress? For this discussion, the rate of progress $\varphi = (f(\vec{x}^t - f(\vec{x}^{t+1}))/f(\vec{x}^t)$ might be defined as the normalized change in the fitness values. Please remind yourself that an evolutionary algorithm is a stochastic optimization procedure, and that you can thus give only statistical, i.e., average, answers.

   Now, keep the number λ of offspring fixed and vary the number μ of parents. How do you think that this affects the rate of progress? Also, please diskuss how you can measure performance/rate of progress?

3. *Practical work:* Validate your theoretical results by running some experiments by using the quadratic fitness function. Modify the `main()`-routine, such that it executes everything from `make_populations()` to `destroy_populations()` a certain number times (e.g., 10 times). In each of these runs, the evolution strategy has to work until the fitness has dropped below 1 % of its initial fitness value. Complete the following two tables. In order to be able to compare you results with those from your classmates, use the following parameter settings: problem size of $n = 100$, initialize all parameters to $x_i = 10$, and use the step size $\sigma = 1$.

$\mu = 1$ parent:

λ	2	3	4	5	6	7	8	10	20	40
generations										
time										

$\lambda = 20$ offspring:

μ	1	2	4	6	8	10
generations						
time						

Questions:

(a) What are the optimal values for μ and λ?

(b) Why didn't you try a (1,1)-evolution strategy? What would its performance be?

4. Finally, you are asked to investigate the difference between a (μ, λ)-evolution strategy and $(\mu + \lambda)$-evolution strategy. Guess, which one will be faster? What are the reasons for your opinion?

   Run a (1,6)-evolution strategy on an $n = 100$ dimensional sphere for 500 generations. Again, use problem size of $n = 100$, initialize all parameters to $x_i = 10$, and use the step size $\sigma = 1$. In every generation, report, i.e., print onto the screen, both the generation number and the parent's fitness value. Save the output into a file. Run the program again but this time a (1+6)-evolution strategy and do the same subsequent steps. Generate a comparative plot. On Linux, you can use `gnuplot` and on Windows, `excel`.

Have fun, Theo and Ralf.

Exercise 9: Evolutionary Theory: Preliminaries

Summer Term 2024

This exercise just looks at some theoretical issues that serve as preliminaries for later theoretical considerations.

Review: Probability and statistics:

1. What is a probability?
2. What is an average? How do you calculate it?
3. What is the standard deviation? How do you calculate it?

To Do: In this exercise you have to do some explorative programming.

Tasks:

1. Write a little program that prints 100 lines each of which should contain the line number and one Gaussian-distributed random number. Plot these numbers (e.g., `gnuplot` in Linux or `excel` in Windows).
2. Reuse the program but print the squares of the random numbers. Do a similar plot.
3. Now, instead of printing 100 lines with a random number or its squares, respectively, print the *sum* of the random numbers and the sum of its squares. Please, complete the following table in which z_i denotes N(0,1)-distributed random numbers:

#	1	4	16	25	100	10000
$\sum z_i$						
$\sum z_i^2$						

   Why does all this happen?

Have fun, Theo and Ralf.

Exercise 10: Convergence

Summer Term 2024

Review: Questions:

1. What does the term *convergence* mean?

2. What does the term *progress* mean?

3. How did we define the term "rate of progress φ"?

4. How is the steepest-descent method defined?

To Do: For the steepest-descent method, you should calculate the rate of progress φ. Furthermore, you should determine the necessary conditions for convergence.

Tasks:

Steepest Descent:

Consider the one-dimensional quadratic fitness function $f(x) = 0.5x^2$.

1. Please calculate the gradient $\nabla f(x) = f^{'}(x)$.
2. Please calculate the rate of progress $\varphi = \ldots$.
3. Under which condition, i.e., values for η, does the steepest-descent method converge?
4. What is the relationship between convergence and the quotient $\zeta = \frac{|x^{t+1}|}{|x^t|}$?
5. Do you need to adapt the step size σ?
6. Please, complete the table and figures presented below for the following values: $\eta = 0.5, \eta = 1.5$, and $\eta = 1.0$. You may always start at $x^{t=0} = 32$.

   What happens for $\eta < 0, \eta = 0, \eta = 2.0$, and $\eta > 2.0$?

   Please, indicate the different regimes in the last figure.

Optional: if you want to do so, generalize your result to the n-dimensional case.

Now, do the same steps for the function $f(x) = 0.25x^4$.

1. Please calculate the gradient $\mathrm{grad} f(x) = \nabla f(x) = f^{'}(x)$.
2. Please calculate the rate of progress $\varphi = \ldots$.
3. Under which condition does the steepest-descent method converge?

4. Do you need to adapt the step size σ?

$\boldsymbol{f(x) = x^2, x^{t=0} = 32, \eta = 0.5}$

	1	2	3	4	5	5	7
x^t							
$f(x^t)$							
grad $f(x^t)$							

$\boldsymbol{f(x) = x^2, x^{t=0} = 32, \eta = 1.5}$

	1	2	3	4	5	5	7
x^t							
$f(x^t)$							
grad $f(x^t)$							

$\boldsymbol{f(x) = x^2, x^{t=0} = 32, \eta = 1.0}$

	1	2	3	4	5	5	7
x^t							
$f(x^t)$							
grad $f(x^t)$							

The different regimes at $x^{t=0} = 32$

Have fun, Theo and Ralf.

Exercise 11: Genetic Algorithms versus Evolution Strategies

Summer Term 2024

The goal of this exercise is to compare genetic algorithms with evolution strategies with respect to some selected runtime issues.

Review: Please, review the following questions from a traditional perspective:

1. What are typical settings for the mutation probability p_m?
2. What are the resulting distributions of the offspring?
3. How do the two types of algorithm handle the step size σ?

To Do: In the following you should explore the possible progress in a graphical as well as analytical way.

Tasks: All the figures below show the same quadratic fitness functions $f(x_1, x_2) = x_1^2 + cx_2^2$, with $c > 1$ denoting an Eigenvalue. The difference is that on the right-hand-sides, the fitness functions are rotated. Such a rotation can be simply obtained by

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} .$$

Assume that all the algorithms start at the little box in the lower left corner.

1. Illustrate the area in which a GA can improve the fitness. Please, consider both cases.
2. Illustrate the area in which an evolution strategy can improve the fitness. Please, do so for both cases.
3. Now, you should consider two parents as inserted in the third figure. Please, indicate the effect of cross over.

Genetic Algorithm

Evolution Strategies

Recombination

Have fun, Theo and Ralf.

Exercise 12: Constructing Simple Threshold Networks

Summer Term 2024

After this exercise, you should be able to construct simple threshold networks.

Review: Please, review the concept of threshold neurons.

1. What is a connection and what the connection weight w_{ij}?
2. What is the netinput net_i?
3. What does the threshold value Θ_i do?
4. How is the threshold unit's transfer function defined?
5. What is the complete definition of a threshold unit?

To Do: For every task, you should summarize the network's logical function in a tabular form, and should then construct a network with a minimal number of threshold units. These exercises require a few think-and-act cycles ◡̈.

Tasks: Please consider the following problems:

1. Logical `and` with three inputs.
2. Logical `or` with four inputs.
3. Simple negation with one input and one output.
4. Logical `nand` with three inputs.
5. An `exclusive-or` function, i.e., (a) or (b) but *not* both, with two inputs and one output.
6. Realize the following two functions:

x_1	x_2	x_3	y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	0

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
1	0	0	1
1	0	1	1
0	1	0	0
0	1	1	0
1	1	0	1
1	1	1	1

Have fun, Theo and Ralf.

Exercise 13: Linear Network Units

Summer Term 2024

Networks that contain only linear units, is another variant of the very many existing network models. After the exercise, you will see why this network model has only educational purposes.

Review: Questions:

1. What are network layers?
2. What does the transfer function do?
3. How is the transfer function in backpropagation multi-layer perceptrons defined?
4. What is the general definition of a network unit?

To Do: Estimate the functionality of linear networks analytically. In this network model, the transfer function is defined as $f(\mathrm{net}_i) = \mathrm{net}_i$.

Tasks: Please, do the following tasks:

1. Define a linear network with two input units, no hidden layer, and one output unit. What is the mathematicl function of this network?
2. What is changing, if you consider two output neurons?
3. Now, please add a hidden layer with two neurons. What is the resulting behavior (functionality) of the resulting network? Please, provide a mathematical description. Hint: it is sufficient to consider only one output unit.
4. What is changing, if you add further hidden layers with one or more hidden units?
5. What is the utility of linear network models?

Have fun, Theo and Ralf.

Exercise 14: The Traditional Perceptron

Summer Term 2024

This exercise covers *the* traditional one-layer perceptron.

Review: Questions:

1. How is the perceptron defined?
2. How can you represent the input/output behavior of a simple perceptron *graphically*?

To Do: Analyse the utility of the perceptron.

Tasks:

1. How does the decision line look like in case of a simple `and` and `or` function? Both functions may have two inputs.
2. We now turn to the classical `xor` case. How does the decision line look like? Is a simple, one-layer perceptron able to realize this function?

   Show by symbolic representations that the simple one-layer perceptron with two inputs x_1 and x_2 as well as a threshold Θ *cannot* realize the `xor` function.
3. In the lecture, we have discussed two realizations of simple, one-layer perceptrons. The first one had a threshold Θ, whereas the other one had an additional bias link with a constant input value of "1". "Suddenly", the second realization has an additional input, and the inequality "$\mathrm{net}_i \geq 0$" means that the angle between the input and weight vector is between $-\pi/2$ and $\pi/2$. Why is this the case? Why may that be the same as a freely moving decision line?

   Hint: This is a surprisingly difficult task, and stop this exercise if you have not succeed within five minutes.

Have fun, Theo and Ralf.

Exercise 15: Quantifying a Network's Quality

Summer Term 2024

In this exercise you should design an error function with which you can measure neural network's mapping quality. This is an essential step towards *learning*.

Review: Please, review the rather "indirect" way of the fitness evaluation in the "Real-World Applications" exercise.

To Do: For the following problems, you should design an appropriate fitness function $E(w_{ij})$, also called error function in the context of neural networks. Please, do the tasks twice, once for units with discrete output values, e.g., threshold units, and once for units with continuous output values.

Tasks: Please consider a neural network with

1. one output unit and one pattern to map

   $E(w_{ij}) =$

   $E(w_{ij}) =$

2. o output units and one pattern to map

   $E(w_{ij}) =$

   $E(w_{ij}) =$

3. one output unit and n patterns to map

   $E(w_{ij}) =$

   $E(w_{ij}) =$

4. o output units and n patterns to map.

   $E(w_{ij}) =$

   $E(w_{ij}) =$

Question:

1. Are your solution the only ones or would you have other options?
2. How would you automate the construction of threshold networks as you have done by hand in the previous exercise?

Have fun, Theo and Ralf.

Exercise 16: Implementation of a Backpropagation Network

Summer Term 2024

It is now time to program your own backpropagation network. The goal is to have a running program with which you can do further exercises. In order to ease this task quite a bit and in order to focus on the algorithmic issues rather than on tedious implementation details we have again prepared a framework for you. Since this exercise still requires some time, you are expected to do this programming as a **homework** and bring the result to the next class!

Review: Please, review the following **questions:**

1. What is the definition of one backpropagation step?

2. What are the main steps and loops in order to do one single backpropagation cycle?

3. What is the definition of the momentum term α?

To Do: Implement a backpropagation network.

Tasks: Happy programming!

1. Download the prepared material from
   `http://www.imd.e-technik.uni-rostock.de/ma/rs/lv/hosc/nn.zip`
2. **Read the API documentation.** You are strongly advised to do so!
3. Implement the main functions in the file `bp.c` of a simple backpropagation network. Please note that the given code material does not provide any backpropagation functionality but demonstrates how to use the network functions. You might want to start with a simple 1-1-1 network that should be realizing an inverter. How many training patterns do you need? This unit is to make sure that your backpropagation code is working well, and the `prt`−functions are mainly for debugging purposes ◡̈. Save this program version for later usage.
4. Add the momentum term α to your program. The required data structures are already included in the programming environment. With $\alpha = 0$, your program should exhibit exactly the same behavior as before. For test purposes, use a small learning rate, e.g., $\eta = 0.1$, and gradually increase the momentum, e.g., from $\alpha = 0, \alpha = 0.1$, to $\alpha = 0.2$. You should be observing an accelerated learning process.

   Question: What is the maximal reasonable value for α?

Have fun, Theo and Ralf.

Exercise 17: Classification with Neural Networks

Summer Term 2024

Pattern *classification* is a typical application area in which neural networks can yield good results. For educational purposes, this exercise resorts to the very simple *encoder* problems. Generally, the classical encoders have n input units, $m = \log_2 n$ hidden units, and n output units. An encoder maps its input patterns onto the same output pattern. Furthermore, any input pattern consists of '0's only, except one single '1'.

Review: What is an appropriate stopping criterion for this task?

To Do: Implement a simple 4-2-4 encode network and explore the influence of both the learning rate η and the momentum α on the speed of the learning process.

Tasks:

1. Please, implement a 4-2-4 encoder that maps the four possible input patterns '1000', '0100', '0010', and '0001' onto identical outputs. Before doing so, discuss the following questions:
   (a) How many training patterns do you have in this application?
   (b) What is a reasonable stopping criterion?
2. Explore how the learning speed depends on the parameter settings for η and α.

α	η					
	10	3	1	0.3	0.1	0.001
0.0						
0.2						
0.4						
0.6						
0.8						
1.0						

Have fun, Theo and Ralf.

Exercise 18: Approximation with Neural Networks

Summer Term 2024

Approximation is another typical application area in which neural networks can yield good results. In this area, a function $f : R^n \rightarrow R^m$ maps an n-dimensional input domain onto an n-dimensional output domain. First, the function is defined by a set of points. Then, a neural network (or any other tool) should learn to approximate the function values in between reasonably well.

Review: Discuss the following questions:

1. What is an appropriate stopping criterion for this task?
2. How many connections w_{ij} does the network need?

To Do: In this exercise, you should implement a neural network that has to learn the simple two-dimensional function $f(x, y) = \cos(x) + \cos(y)$, i.e., $f : R^2 \rightarrow R$, in the range $(x, y) \in [-\pi..\pi]$.

Questions:

1. How many input and output units do you need?
2. What is the output range of a neuron, if you use the regular logistic transfer function?
3. How/where can you modify the number of network parameters?

Tasks: Reuse your simple backpropagation network that you have implemented in the previous exercise. Strip of the encoder stuff and loosely follow the following steps:

1. Introduce two parameters `l_pats` and `t_pats` that specify the number of learning and test patterns *per* input dimension. How many patterns do you need in total?
2. Introduce another parameter that specifies whether the training and test pattern should be generated randomly or systematically.
3. Complete the program and print both the learning and the test error.

   Hint: The documentation shows how you can directly call `gnuplot` from your own program.
4. Try to learn and generalize the given function reasonably well. In so doing, vary the learning rate η, the momentum α, the number of network parameters, the number of training patterns, the initialization mode, and the number of learning cycles. What can you observe?

Have fun, Theo and Ralf.