Artificial Neural Networks

Prof. Dr. Sen Cheng

Oct 14, 2019

Problem Set 2: Optimization

Tutors: Filippos Panagiotou (filippos.panagiotou@ini.rub.de), Mohammad M. Nejad (m.mohagheghi@ini.rub.de)

1. **Analytical minimization of a function**

   Study the minima of the function
   $$L(x) = x^4 - 4x^2 + 4 \tag{1}$$
   by following the steps below.

   (a) Find the roots of $\frac{\mathrm{d}}{\mathrm{d}x}L(x)$.

   (b) Determine whether the roots of the derivative are minima or maxima of $L(x)$. What is the sufficient condition for minima and maxima?

   (c) Plot L against x in Python for $x \in [-2,2]$ and mark the minima (in red) and maxima (in green) in the plot.

2. **Numerical minimization of a function**

   Use gradient descent to find the minima of $L(x)$ from the problem above.

   (a) Implement gradient descent to find the minima of $L(x)$.

   (b) Starting the estimatation algorithm with different values, $x_0 \in [-2,2]$, run the gradient descent algorithm with $\eta = 0.01$ and $N = 50$ (N, number of iterations). Observe how the starting value determines which local minimum is found.

   (c) Set your initial guess $x_0 = 1$ and $N = 100$, use different learning rates $\eta \in [0.05, 0.15]$ (use `np.linspace`) to find the minimum. Calculate the precision of the estimation by computing the absolute value of the difference between the estimated and the actual minimum for each η. Plot the precision against η. How does η affect the precision of the estimation?

   (d) Now set your initial guess $x_0 = 1$ and $\eta = 0.01$. Use different number of iterations, $N \in [1,100]$. How does N affect the precision of the estimation?

3. **The role of the learning rate**

   To better understand the implications of choosing an appropriate learning rate η in gradient descent, let us focus on a simpler loss function $V(x) = x^2$, because it has only one minimum.

   (a) Use gradient descent to find the minimum of this function. To better understand the method you should keep track of the estimated minimum in each iteration, find the corresponding point on the function $V(x)$ and plot them together with the loss function.

   (b) Set your initial guess $x_0 = 1$ and try few different $\eta \in [0.1, 1.1]$. Observe how the estimate depends on the learning rate. Find at least one η for which the estimate converges and one η for which it diverges.

Solutions

1. **Analytical minimization of a function**

   (a) $\frac{\mathrm{d}}{\mathrm{d}x}L(x) = 0$:

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}x}L(x) &= 4x^3 - 8x = 4x(x^2-2) = 4x(x-\sqrt{2})(x+\sqrt{2}) \overset{!}{=} 0 \\
x &= -\sqrt{2}, 0, \sqrt{2}
\end{aligned}
$$

   (b) To distinguish minima (desired in minimizing cost functions) from maxima we must look at the second derivative $\frac{\mathrm{d}^2}{\mathrm{d}x^2}L(x)$ at x's where $\frac{\mathrm{d}}{\mathrm{d}x}L(x) = 0$. For the minima $\frac{\mathrm{d}^2}{\mathrm{d}x^2}L(x) > 0$ and for maxima $\frac{\mathrm{d}^2}{\mathrm{d}x^2}L(x) < 0$.

$$
\begin{aligned}
\frac{\mathrm{d}^2}{\mathrm{d}x^2}L(x) &= L''(x) = 12x^2 - 8 \\
L''(-\sqrt{2}) &= 16 > 0 \text{ (minimum)} \\
L''(0) &= -8 < 0 \text{ (maximum)} \\
L''(\sqrt{2}) &= 16 > 0 \text{ (minimum)}
\end{aligned}
$$

   (c) The visualization of the loss function and its minima and maximum (Fig. 1).

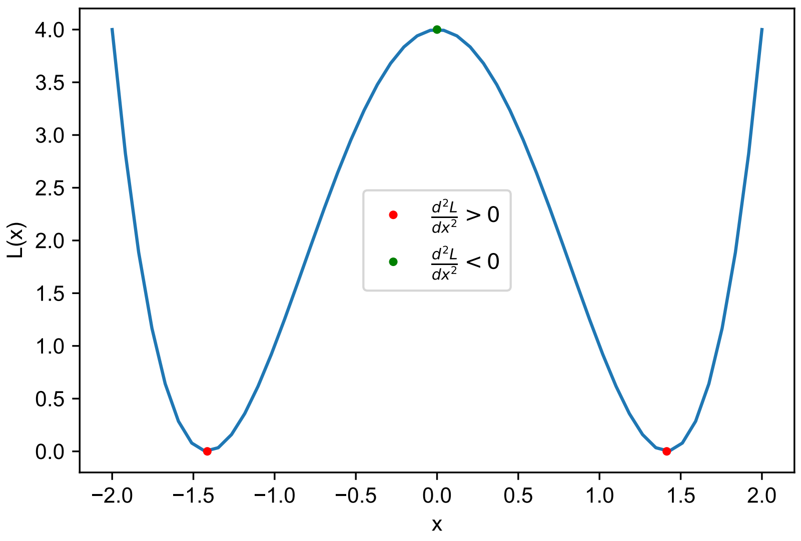

Figure 1: The loss function $L(x)$ is in blue, the minima are in red and the maximum is in green.

```
# Python packages required for this assignment

import numpy as np
import matplotlib.pyplot as plt

# Declared functions

def L(x):              #Cost function
    return x**4-4*x**2+4
```

```
def d2L(x):          #Second derivative of the cost function
    return 12*x**2-8


# Costumizing the fonts used in plots
plt.rcParams['font.family'] = 'sans-serif'
plt.rcParams['font.sans-serif'] = 'Arial'
plt.rcParams['font.size'] = 10

x = np.linspace(start=-2, stop=2, num=50)
l = L(x)

fig, ax = plt.subplots()

#roots of the first derivative
root_dldx = np.array([-np.sqrt(2), 0, np.sqrt(2)])
#Points on L(x) that corresponds to the roots of the first derivative of L
lroot = L(root_dldx)

ax.plot(x, l)
ax.set_xlabel('x')
ax.set_ylabel('L(x)')
d2l = d2L(root_dldx)
ax.plot(root_dldx[d2l>0], lroot[d2l>0], '.r',
        label=r'$\frac{d^2L}{dx^2}>0$')
ax.plot(root_dldx[d2l<0], lroot[d2l<0], '.g',
        label=r'$\frac{d^2L}{dx^2}<0$')
ax.legend()
# Saving the figure
fig.savefig('a2-e1-lossfunction-extrema.pdf',
            format='pdf')
```

2. **Numerical minimization of a function**

   (a)

```
import numpy as np

def grad_descent(l_rate, num_iter, init_min, dL):
    '''
    Inputs
    l_rate: learning rate
    num_iter: number of iterations
    init_guess: initial guess for the minimum
    dL: derivative of the loss function

    Outputs
    mins: An array of the estimated minimum for each iteration.
          mins[-1] gives the final estimation.
    '''
    mins = init_min*np.ones(num_iter+1)
    for it in range(num_iter):
        min_ = init_min - l_rate*dL(init_min)
        init_min = min_
        mins[it+1] = min_
    return mins
```

(b) Gradient descent can only find one minimum based on the initial guess that was chosen. When the initial guess is negative gradient descent finds the leftmost minimum and when the guess is positive gradient descent finds the rightmost minimum (Fig. 2).

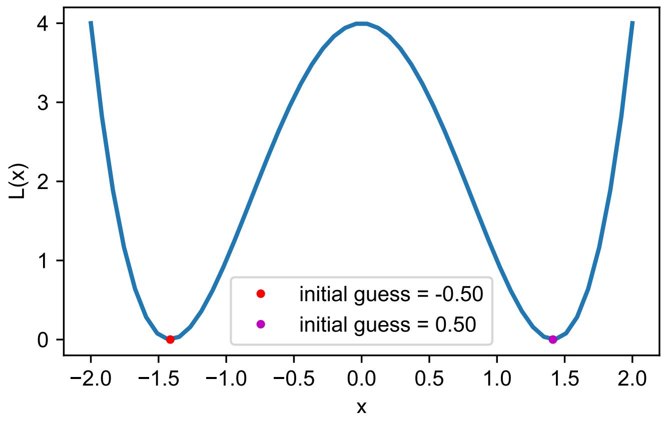

Figure 2: The role of the the initial guess on the estimation of the minimum.

(c) For the given learning rate (η) and initial value, as η increases the estimation becomes less precise (Fig. 3a).

(d) For the given number of iterations (N) and initial value, as N increases the estimation becomes more precise (Fig. 3b).

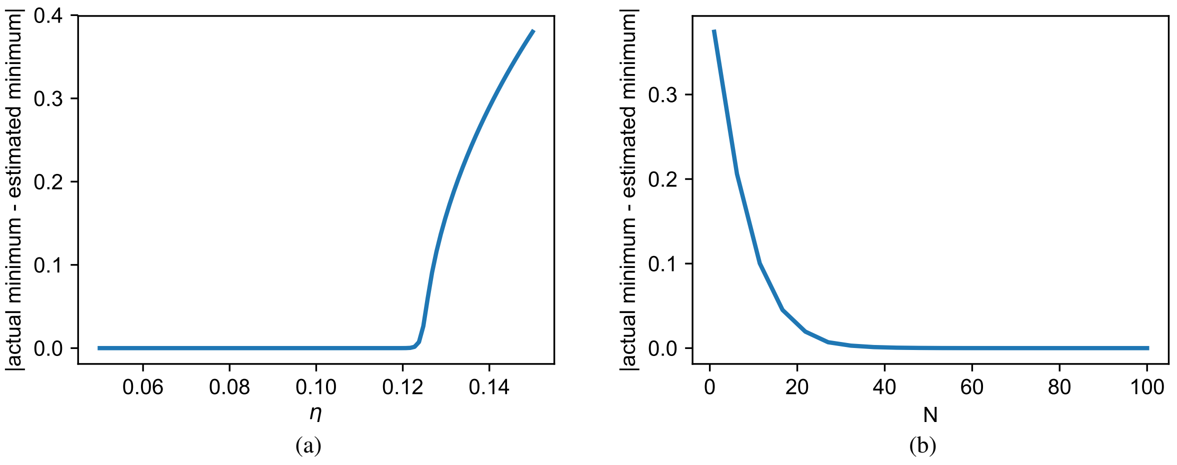

Figure 3: The role of η (a) and N (b) on the precision of the estimation.

```
# Python packages required for this assignment

import numpy as np
import matplotlib.pyplot as plt
```

```
def L(x):                          #Loss function
    return x**4-4*x**2+4

def dLdx(x):                       #Analytical derivative of loss function
    return 4*x**3-8*x

def dL_num(x):     #Numerical implementation of derivation
    dx=1e-4#Precision of neighboring point for computation of the derivative
    dif = (L(x+dx)-L(x))/dx
    return dif

def grad_descent(l_rate, num_iter, init_min, dL):
    mins = init_min*np.ones(num_iter+1)
    for it in range(num_iter):
        min_ = init_min - l_rate*dL(init_min)
        init_min = min_
        mins[it+1] = min_
    return mins


# Costumizing the fonts used in plots
plt.rcParams['font.family'] = 'sans-serif'
plt.rcParams['font.sans-serif'] = 'Arial'
plt.rcParams['font.size'] = 10

'''
PART B
'''

init_min = -.5              #Initial guess
learning_rate = 0.01
num_iter = 50               #Number of iterations

est_min = grad_descent(learning_rate, num_iter, init_min, dLdx)
min_ = est_min[-1]
l_min = L(min_)

x = np.linspace(start=-2, stop=2, num=50)
l = L(x)

fig, ax = plt.subplots(figsize=(5,3))

ax.plot(x, l, linewidth=2)
ax.set_xlabel('x')
ax.set_ylabel('L(x)')
ax.plot(min_, l_min, '.r', label='initial guess = %.2f' %init_min)
ax.legend()


init_min = .5               #Initial guess

est_min = grad_descent(learning_rate, num_iter, init_min, dLdx)
```

```
min_ = est_min[-1]
l_min = L(min_)

ax.plot(min_, l_min, '.m', label='initial guess = %.2f' %init_min)
ax.legend()
fig.savefig('a2-e2-initguess.pdf', format='pdf')


'''
PART C
'''

init_min = 1            #Initial guess
learning_rate = np.linspace(0.05, 0.15, 100)
num_iter = 100          #Number of iterations
mins = np.zeros_like(learning_rate)

for ind, l_ in enumerate(learning_rate):
    est_min = grad_descent(l_, num_iter, init_min, dLdx)
    mins[ind] = est_min[-1]

fig, ax = plt.subplots(figsize=(4,3))
ax.plot(learning_rate, np.abs(mins-np.sqrt(2)), linewidth=2)
ax.set_xlabel(r'$\eta$')
ax.set_ylabel('|actual minimum - estimated minimum|')
fig.savefig('a2-e2-error-lrate.pdf', format='pdf')

'''
PART D
'''

init_min = 1            #Initial guess
learning_rate = 0.01
num_iters = np.linspace(1,100, 20)          #Number of iterations
mins = np.zeros_like(num_iters)

for ind, num_iter in enumerate(num_iters):
    est_min = grad_descent(learning_rate, int(num_iter), init_min, dLdx)
    mins[ind] = est_min[-1]

fig, ax = plt.subplots(figsize=(4,3))
ax.plot(num_iters, np.abs(mins-np.sqrt(2)), linewidth=2)
ax.set_xlabel('N')
ax.set_ylabel('|actual minimum - estimated minimum|')
fig.savefig('a2-e2-error-N.pdf', format='pdf')
```

3. **The role of the learning rate**

   (a) The minimum of $V(x)$ is 0 (Fig. 4).

   (b) When $\eta = 1.1$ the estimate diverges and in this case more iterations will make the estimate worse (Fig. 5a). When $\eta = 0.1$ the estimate converges, but the number of iterations in this case, $N = 4$, is not sufficient for

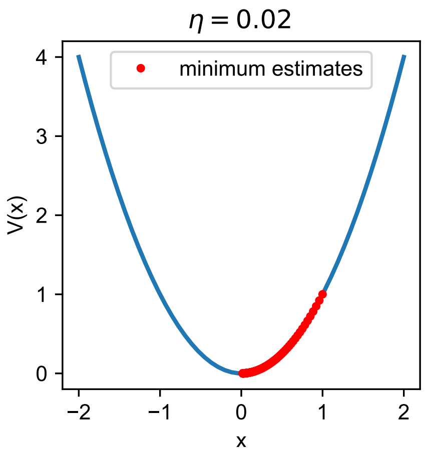

Figure 4: The estimate gets closer and closer as the gradient descent reaches its last iterations.

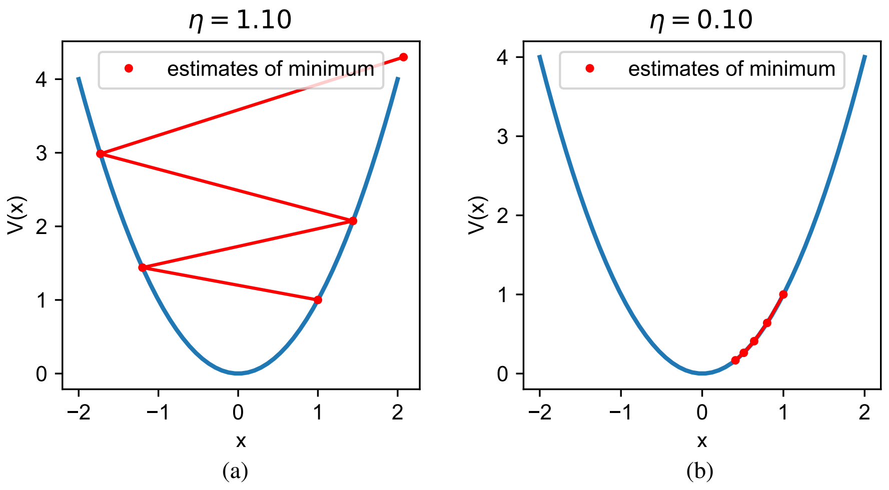

Figure 5: Diverging (a) and converging (b) estimates.

the estimate to reach the minimum (Fig. 5b).

```
import numpy as np
import matplotlib.pyplot as plt

def grad_descent(l_rate, num_iter, init_min, dL):
    mins = init_min*np.ones(num_iter+1)
    for it in range(num_iter):
        min_ = init_min - l_rate*dL(init_min)
        init_min = min_
        mins[it+1] = min_
    return mins

def V2(x):                    #The simple loss function
    return x**2

def dV2(x):
    return 2*x

def dV2_num(x):    #Numerical implementation of derivation
```

```
    dx=1/10000#Precision of neighboring point for computation of the derivative
    dif = (V2(x+dx)-V2(x))/dx
    return dif

def regressor(a, x):
    return a*x

def loss(y, a, x):
    return np.sum(np.abs(y - regressor(a, x)))

def plot_grad_descent(learning_rate, num_iter, init_min, filename):
    min_vec = grad_descent(learning_rate, num_iter, init_min, dV2)
    x = np.linspace(start=-2, stop=2, num=50)

    fig, ax = plt.subplots(figsize=(3,3))

    ax.plot(x, V2(x), linewidth=2)
    ax.plot(min_vec, V2(min_vec), color='red')
    ax.plot(min_vec, V2(min_vec), '.r', label='minimum estimates')
    ax.set_ylabel('V(x)')
    ax.set_xlabel('x')
    ax.set_title(r'$\eta=%.2f$' %learning_rate)
    ax.legend()
    fig.savefig(filename, format='pdf')

# Costumizing the fonts used in plots
plt.rcParams['font.family'] = 'sans-serif'
plt.rcParams['font.sans-serif'] = 'Arial'
plt.rcParams['font.size'] = 10


# PART A
plot_grad_descent(0.02, 100, 1, 'a2-e3-minimum.pdf')

# PART B
plot_grad_descent(1.1, 4, 1, 'a2-e3-diverging.pdf')
plot_grad_descent(.1, 4, 1, 'a2-e3-converging.pdf')
```