Optimization and Computational Linear Algebra for Data Science
Homework 12: Gradient descent

Problem 12.1 (2 points). **(a)** *f has a local maximum at $(1.2, 1.3)$ and a global maximum at $(-0.5, -0.7)$. f has a local minimum at $(-0.9, 0.7)$ and a global minimum at $(0.9, -0.9)$ and a saddle-point at $(0.9, 0)$.*

(b) *When initialized at A, gradient descent is likely to converge to the local minimum at $(-0.9, 0.7)$. When initialized at B, gradient descent is likely to converge to the global minimum at $(0.9, -0.9)$.*

Problem 12.2 (5 points).

$$f(x) = \frac{1}{2} x^{\mathsf{T}} M x - \langle x, b \rangle + c$$

(a) *Let $x \in \mathbb{R}^d$. f is twice differentiable and*

$$H_f(x) = M.$$

By definition of μ and L, the eigenvalues of M are all above μ and all smaller than L: f is therefore μ-strongly convex and L-smooth. f is therefore convex. Hence

$$\begin{aligned} x \text{ is a global minimizer of } f \quad &\iff \quad \nabla f(x) = 0 \\ &\iff \quad Mx - b = 0 \\ &\iff \quad x = M^{-1} b. \end{aligned}$$

$x^* = M^{-1}b$ is therefore the unique global minimizer of f.

(b) *$\nabla f(x) = Mx - b$ hence*

$$\begin{aligned} x_{t+1} - x^* &= x_t - x^* - \beta(Mx_t - b) = x_t - x^* - \beta M(x_t - M^{-1}b) \\ &= x_t - x^* - \beta(x_t - x^*) \\ &= (\mathrm{Id} - \beta M)(x_t - x^*). \end{aligned}$$

(c) *Let $B = \mathrm{Id} - \beta M$. B is symmetric and his eigenvalues are:*

$$1 - \lambda_1/L, \dots, 1 - \lambda_d/L$$

which are all between 0 and $1 - \mu/L$. The largest eigenvalue of B^2 is therefore $(1 - \mu/L)^2$. Since the singular values of B are the square root of the eigenvalues of $B^{\mathsf{T}}B = B^2$ because B is symmetric, we get that the largest singular value of B is $1 - \mu/L$.

We know that the spectral norm of a matrix is equal to its largest singular value: $\|B\|_{\mathrm{Sp}} = 1 - \mu/L$. Hence

$$\|x_{t+1} - x^*\| = \|B(x_t - x^*)\| \leq \|B\|_{\mathrm{Sp}} \|x_t - x^*\| = \Big(1 - \frac{\mu}{L}\Big) \|x_t - x^*\|,$$

from which the result follows.

(d) *Since* $w_{t+1} = (\mathrm{Id} - L^{-1}M)w_t$, *we have for* $i \in \{1, \dots, d\}$

$$\alpha_i(t+1) = v_i^\top (\mathrm{Id} - L^{-1}M)w_t = (v_i^\top - L^{-1}v_i^\top M)w_t.$$

Now, we use the fact that $Mv_i = \lambda_i v_i$ *to get* $v_i^\top M = \lambda_i v_i^\top$*:*

$$\alpha_i(t+1) = (1 - \lambda_i/L)\alpha_i(t).$$

This gives

$$\alpha_i(t) = (1 - \lambda_i/L)^t \alpha_i(0).$$

(e) *Let* $i \in \{1, \dots, d\}$. $|\alpha_i(t)| = |\langle v_i, x_t - x^* \rangle|$ *is equal to the norm of the orthogonal projection of* $x_t - x^*$ *onto* $\mathrm{Span}(v_i)$*, that is corresponds to «the distance between* x_t *and* x^* *in the direction of* v_i*».*

From the previous we see that at each iteration of gradient descent, this «distance» is multiplied by a factor $1 - \lambda_i/L$*, where* $_i \in [\mu, L]$*. Hence, gradient descent converges faster «in the direction of* v_i*» if* λ_i *is large (close to* L*).*

(f) $(\alpha_1(t), \dots, \alpha_d(t))$ *are the coordinates of* $w_t = x_t - x^*$ *in the orthonormal basis* $(v_1, \dots, v_d)$*. Therefore*

$$\|x_t - x^*\| = \sqrt{\sum_{i=1}^{d} \alpha_i(t)^2} = \sqrt{\sum_{i=1}^{d} \Big(1 - \frac{\lambda_i}{L}\Big)^{2t} \big\langle v_i, x_0 - x^* \big\rangle^2}.$$

Probleme 12.4.

$$x_{t+1} = x_t - \beta(Mx_t - b) + \gamma x_t - \gamma x_{t-1}$$

$$= ((1+\gamma)\mathrm{Id} - \beta M)\, x_t + \beta b - \gamma x_{t-1}$$

Subtracting $x^* = M^{-1}b$ on both sides gives:

$$x_{t+1} - x^* = ((1+\gamma)\mathrm{Id} - \beta M)(x_t - x^*) - \gamma(x_{t-1} - x^*)$$

$(\alpha_1(t), \dots \alpha_d(t))$ are the coordinates of $x_t - x^*$ in the orthonormal basis $(v_1 \dots v_d)$. Hence

$$\alpha_i(t+1) = \langle v_i, x_{t+1} - x^* \rangle$$
$$= v_i^T((1+\gamma)\mathrm{Id} - \beta M)(x_t - x^*) - \gamma \alpha_i(t-1)$$
$$= (1+\gamma - \beta\lambda_i)\, \alpha_i(t) - \gamma \alpha_i(t-1)$$

We have to study the order-2 recursion
$\alpha_i(t+1) - (1+\gamma-\beta\lambda_i)\alpha_i(t) + \gamma\alpha_i(t-1)$.

To do so, we have to compute the roots of $X^2 - (1+\gamma-\beta\lambda_i)X + \gamma$.

The discriminant of this quadratic function is

$$\Delta = (1+\gamma-\beta\lambda_i)^2 - 4\gamma.$$

$$1+\gamma-\beta\lambda_i = \frac{(\sqrt{L}+\sqrt{\mu})^2 + (\sqrt{L}-\sqrt{\mu})^2 - 4\lambda_i}{(\sqrt{L}+\sqrt{\mu})^2}$$

$$= \frac{2}{(\sqrt{L}+\sqrt{\mu})^2}\left(L+\mu-2\lambda_i\right)$$

$$\Delta = \frac{4}{(\sqrt{L}+\sqrt{\mu})^4}\left((L+\mu-2\lambda_i)^2 - (\sqrt{L}-\sqrt{\mu})^2(\sqrt{L}+\sqrt{\mu})^2\right)$$

$$= \frac{4}{(\sqrt{L}-\sqrt{\mu})^4}\left((L+\mu-2\lambda_i)^2 - (L-\mu)^2\right)$$

$$= \frac{4}{(\sqrt{L}-\sqrt{\mu})^4}\left((2\mu-2\lambda_i)(2L-2\lambda_i)\right)$$

$$= \frac{16}{(\sqrt{L}-\sqrt{\mu})^4}(\mu-\lambda_i)(L-\lambda_i) \leq 0.$$

The roots of $X^2-(1+\gamma-\beta\lambda_i)X+\gamma$ are therefore:

$$x = \frac{1}{2}\left((1+\gamma-\beta\lambda_i) + i\sqrt{|\Delta|}\right) \text{ and } x' = \frac{1}{2}\left((1+\gamma-\beta\lambda_i) - i\sqrt{|\Delta|}\right)$$

There exists two constants $C_1, C_2 \in \mathbb{R}$ such that for all $t \geq 0$:

$$\alpha_i(t) = C_1 x^t + C_2 (x')^t$$

Compute

$$|x| = |x'| = \frac{1}{2}\sqrt{(1+\gamma-\beta\lambda_i)^2 - \Delta} = \sqrt{\gamma} = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$$

Hence $|\alpha_i(t)| \leq (|C_1|+|C_2|)\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^t$

We conclude that

$$\|x_t - x^*\|^2 = \sum_{i=1}^{d} \alpha_i(t)^2$$

$$\leq \underbrace{\left(\sum_{i=1}^{d} c_i^2\right)}_{\overset{\text{def}}{=} C^2} \left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^{2t}$$

Hence $\|x_t - x^*\| \leq C \left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^{t}$

In [1]:

```
from mpl_toolkits import mplot3d
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
plt.rc('font',family='serif')
```

In [2]:

```
d=1000
n=2000
A = np.random.normal(size=(n,d)) / np.sqrt(n)
y = np.random.normal(size=n)
lambd= 1
```

We consider the Ridge cost:

$$f(x) = \frac{1}{2}\|Ax - y\|^2 + \frac{\lambda}{2}\|x\|^2,$$

where $\lambda > 0$ is some regularization parameter.

(a) Show that f is can be written in the format the function f of Problem 12.2, for some $M \in \mathbb{R}^{d \times d}$, $b \in \mathbb{R}^d$ and $c \in \mathbb{R}$. Compute numerically the values of L and μ. Plot the eigenvalues of $H_f(x)$ using an histogram.

We have

$$f(x) = \frac{1}{2}\|Ax - y\|^2 + \frac{\lambda}{2}\|x\|^2 = \frac{1}{2}x^T A^T Ax - \langle x, A^T y\rangle + \frac{1}{2}\|y\|^2 + \frac{\lambda}{2}\|x\|^2 = \frac{1}{2}x^T(A^T A + \lambda Id)x -$$

Hence f can be put in the format of Problem 12.2 with $M = A^T A + \lambda Id$, $b = A^T y$ and $c = \frac{1}{2}\|y\|^2$.

In [32]:

```
M = A.T @ A + lambd * np.identity(d)
w,v = np.linalg.eigh(M)
L=np.max(w)
mu=np.min(w)
print('L is equal to ',L ,'and mu is equal to ',mu)
```

```
L is equal to  3.9042013011704997 and mu is equal to  1.0850698634375
893
```

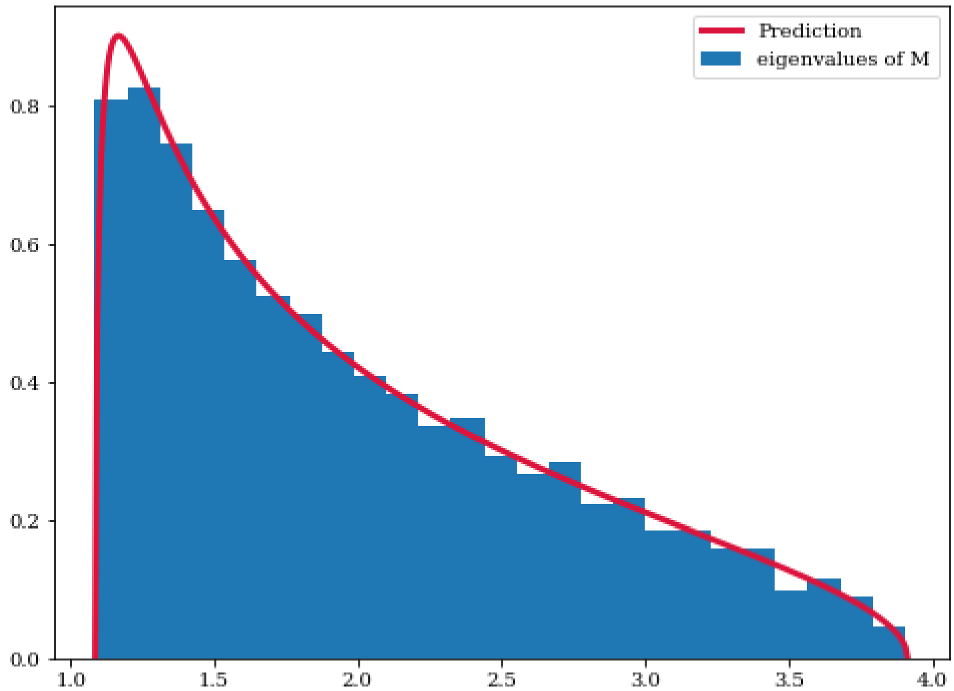
In [31]:

```
# Optional: the limiting shape of the histogram of the eigenvalues of M
# is known as the "Marcenko-Pastur" distribution. We plot is below

lambdaMin = (1-np.sqrt(d/n))**2
lambdaMax = (1+np.sqrt(d/n))**2
print(lambdaMin+lambd,lambdaMax+lambd)

t=np.linspace(lambdaMin,lambdaMax,400)
MP = (1/(2*np.pi))*(n/d)*np.sqrt((lambdaMax-t)*(t-lambdaMin))/t

plt.figure(figsize=(8,6))
plt.hist(w,bins=25,density=True, label='eigenvalues of M')
plt.plot(t+lambd,MP,color='crimson',linewidth=3, label='Prediction')
plt.legend()
```

```
1.0857864376269049 3.914213562373095
```

Out[31]:

```
<matplotlib.legend.Legend at 0x7f343ce14400>
```



(b) Implement gradient descent with constant step-size $\beta = 1/L$ (as in Problem 12.2), with random initial position x_0. Plot the log-error $\log(\|x_t - x_*\|)$ as a function of t.

In [33]:

```
T=30
B = np.identity(d) - M/L
b = A.T @ y
x=np.random.normal(size=(d,T))
for t in range(T-1):
    x[:,t+1] = B @ x[:,t] + b /L
```

In [34]:

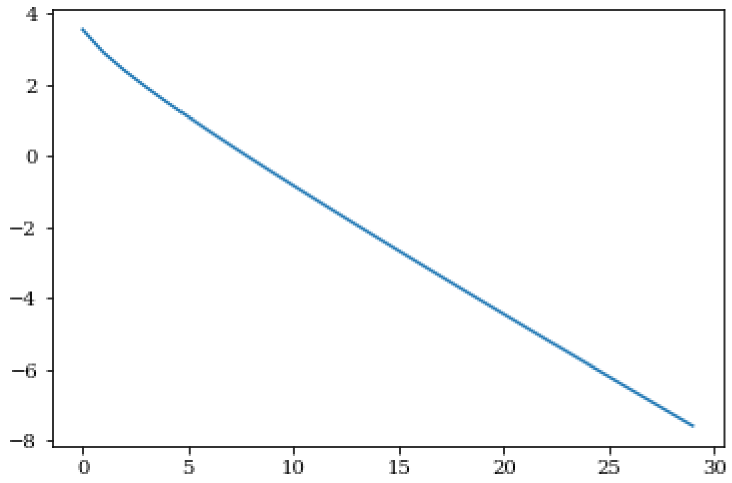
```
x_star= np.linalg.inv(M) @ A.T @ y
```

In [35]:

```
error = [np.sqrt(np.sum(np.square(x[:,t]-x_star))) for t in range(T)]
plt.plot(np.arange(T),np.log(error))
```

Out[35]:

```
[<matplotlib.lines.Line2D at 0x7f3440d7b280>]
```



(c) Implement gradient descent with momentum, with the same parameters as in Problem 12.4. Plot the log-error $\log(\|x_t - x_*\|)$ as a function of t, on the same plot than the log-error of gradient descent without momentum.

In [36]:

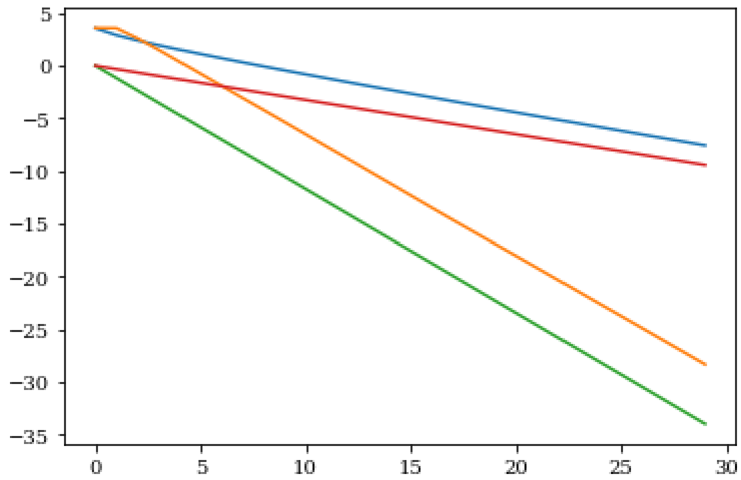
```
T=30
gamma = ((np.sqrt(L)-np.sqrt(mu))/(np.sqrt(L)+np.sqrt(mu)))**2
beta= 4 / (np.sqrt(mu)+np.sqrt(L))**2
B = np.identity(d) - beta*M
b = A.T @ y
x=np.random.normal(size=(d,T))
for t in range(T-2):
    x[:,t+2] = B @ x[:,t+1] + beta*b + gamma*(x[:,t+1]-x[:,t])
```

In [37]:

```
error2 = [np.sqrt(np.sum(np.square(x[:,t]-x_star))) for t in range(T)]
plt.plot(np.arange(T),np.log(error))
plt.plot(np.arange(T),np.log(error2))
t=np.arange(T)
plt.plot(t*np.log(gamma)/2)
plt.plot(t*np.log(1- mu/L))
```

Out[37]:

```
[<matplotlib.lines.Line2D at 0x7f3440d47730>]
```



In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```