Problem 1

March 17, 2023

[1]:
```
using Plots
using AdvConvex.HW3
using AdvConvex.HW4
using Optim
```

```
[ Info: Precompiling Plots
[91a5bcdd-55d7-5caf-9e0b-520d859cae80]
[ Info: Precompiling AdvConvex
[a70558b1-94d0-46ca-a15d-76cbf33c1d08]
[ Info: Precompiling Optim
[429524aa-4258-5aef-a3af-852621145aeb]
```

[2]:
```
mat = get_spam_data()
X_train, Y_train, X_test, Y_test = train_test_split(mat, 0.05)
```

[2]:
```
([-2.3025850929940455 -1.7147984280919266 … -2.3025850929940455
-2.3025850929940455; -2.3025850929940455 -2.3025850929940455 …
-2.3025850929940455 -2.3025850929940455; … ; 2.4932054526026954 3.7864597824528
… 2.7788192719904172 2.4932054526026954; 4.883559211528279 6.499937405290376 …
4.11251186617755 4.160444363926624], [1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0,
1.0, 1.0, -1.0  …  -1.0, -1.0, -1.0, -1.0, 1.0, 1.0, -1.0, 1.0, 1.0, 1.0],
[-2.3025850929940455 -2.3025850929940455 … -0.030459207484708574
-1.3862943611198906; -2.3025850929940455 -2.3025850929940455 …
-1.3093333199837622 -1.6094379124341003; … ; 2.4069451083182885
1.9600947840472698 … 4.763028270603671 3.893859034800475; 3.7864597824528
2.7788192719904172 … 8.159975242934362 6.933520486868163], [1.0, -1.0, -1.0,
-1.0, -1.0, -1.0, -1.0, 1.0, -1.0, 1.0  …  -1.0, -1.0, 1.0, -1.0, 1.0, 1.0,
-1.0, -1.0, 1.0, -1.0])
```

[4]:
```
f = LogRegProblem(X_test,Y_test)
 f(w) = HW3. (f, w)

prob = DifferentiableProblem(f,  f)
nest_solver = NesterovDescentSolver(
      = 1e-2,
      = 0.0,
    max_iter = 10^4,
    linesearch = BackTrackingLineSearch(),
)

w_opt_nest, hist_nest = HW4.solve(nest_solver, prob, zeros(size(X_test, 1)));
```

[5]:

```
gd_solver = GradientDescentSolver(
   = 1e-3,
   = 1e-10,
    max_iter = 10^4,
    linesearch = BackTrackingLineSearch(),
)
w_opt_gd, hist_gd = HW3.solve(gd_solver, prob, zeros(size(X_test, 1)));
```

[6]:

```
res = optimize(f, zeros(size(X_test, 1)), NelderMead(),
    Optim.Options(iterations=10_000, show_trace=false, store_trace=true)
)
```

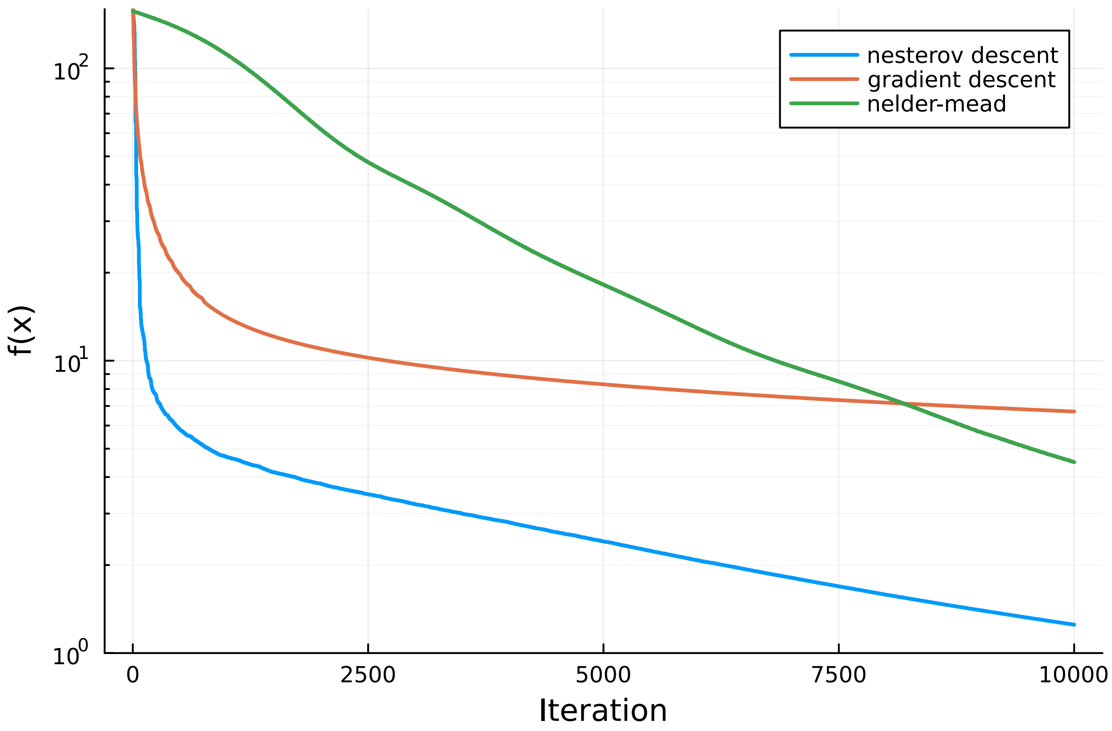
[6]:

```
 * Status: failure (reached maximum number of iterations)

 * Candidate solution
    Final objective value:     4.497449e+00

 * Found with
    Algorithm:     Nelder-Mead

 * Convergence measures
    √(Σ(yᵢ-ȳ)²)/n   1.0e-08

 * Work counters
    Seconds run:   0  (vs limit Inf)
    Iterations:    10000
    f(x) calls:    13724
```

[7]:

```
plot(
    hist_nest.f, yscale=:log10,
    label="nesterov descent", lw=2,
    xlabel="Iteration", ylabel="f(x)",
    ylims=(10^(floor(log10(last(hist_nest.f)))),Inf), yminorgrid=true)
plot!(hist_gd.f, label="gradient descent", lw=2)
plot!(getfield.(res.trace, :value), label="nelder-mead", lw=2)
```

[7]:



nesterov descent
gradient descent
nelder-mead
f(x)
10^2
10^1
10^0
0
2500
5000
7500
10000
Iteration

Problem 2

March 17, 2023

[73]:

```
using Plots
using AdvConvex.HW3
using AdvConvex.HW4
using Optim
using LinearAlgebra
# NOTE: nbconvert doesn't render convenient unicode stuff like lambdas and nablas
```

[74]:

```
mat = get_spam_data()
X_train, Y_train, X_test, Y_test = train_test_split(mat, 0.334);
```

[90]:

```
f = LogRegProblem(X_train,Y_train)
 f(x) = HW3. (f,x)
prob = DifferentiableProblem(f,  f)
solver = GradientDescentSolver(
      = 1e-4,
      = 0.0,
    max_iter=5_000,
    linesearch = BackTrackingLineSearch()
)
w_opt1, hist1 = solve(solver, prob, zeros(size(X_train, 1)));
```

[91]:

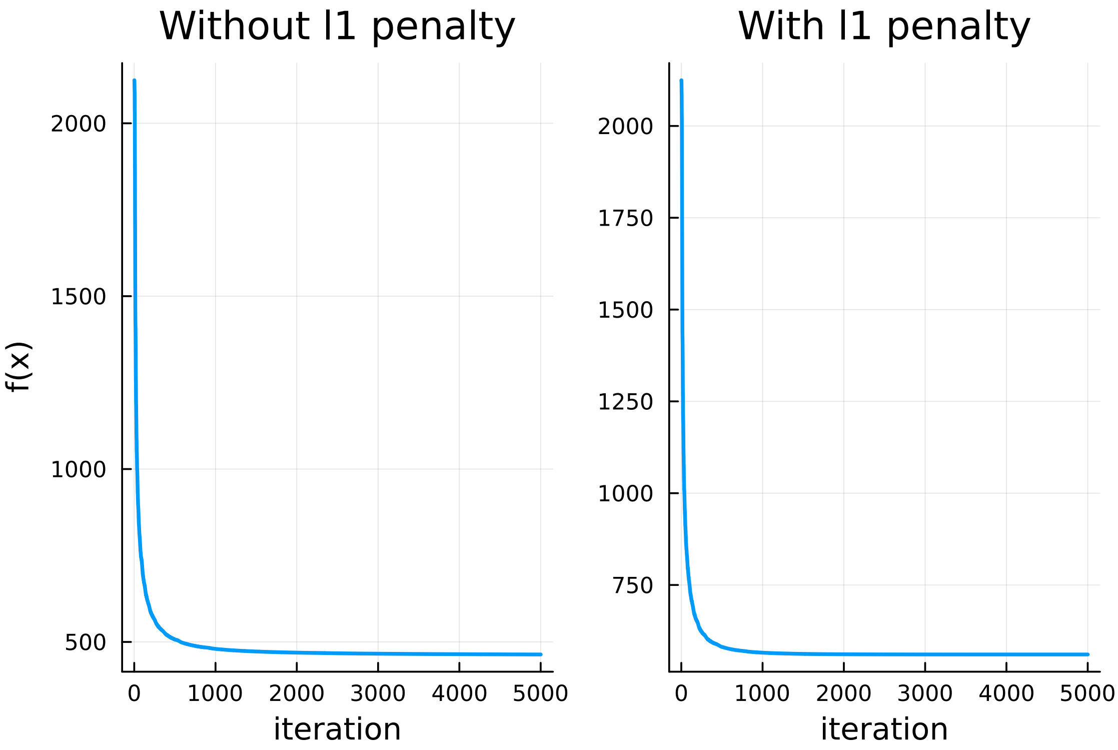
```
  = 5.0
l = PenaltyLogRegProblem(f,  )

g(l::PenaltyLogRegProblem,w) = l.logreg(w)
 g(l,w) = HW3. (l.logreg, w)
h(l::PenaltyLogRegProblem,w) = l.  * norm(w, 1)
loss(l::PenaltyLogRegProblem, w) = g(l,w) + h(l,w)
prox_th(l::PenaltyLogRegProblem, t, y) = sign(y)*max(abs(y) - t*l. , 0.0)

p = ProximalProblem(
    w -> loss(l, w),
    w ->  g(l, w),
    (y,t) -> HW4.prox_th(l,t,y)
)
```

```
w0 = zeros(size(X_test, 1))
solver = GradientDescentSolver(
      = 1e-4,
      = 0.0,
    max_iter=5_000,
    linesearch = BackTrackingLineSearch()
)

w_opt2, hist2 = HW4.solve(solver, p, zeros(size(X_train, 1)));
```

[92]:

```
plot(
    plot(hist1.f, label="",lw=2,ylabel="f(x)",xlabel="iteration",␣
 ↪title="Without l1 penalty"),
    plot(hist2.f,label="",lw=2,xlabel="iteration", title="With l1 penalty")
)
```

[92]:



[94]:

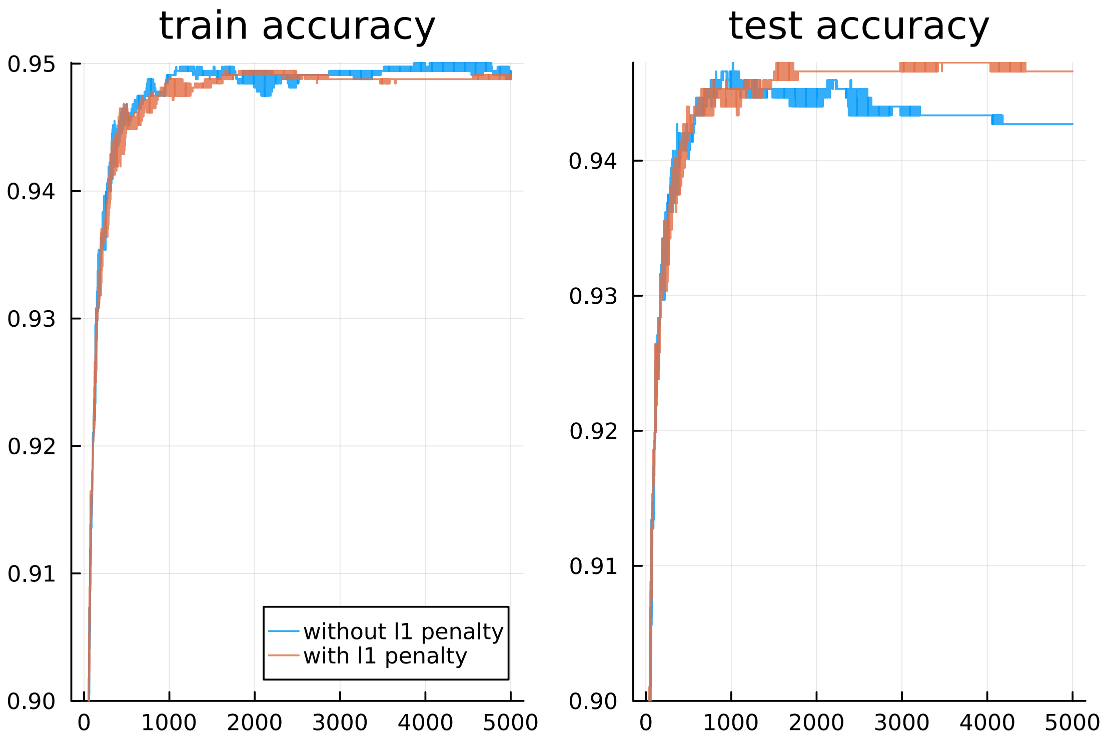
```
test_acc1 = map(hist1.x) do x
    HW3.accuracy(x, X_test, Y_test)
end

test_acc2 = map(hist2.x) do x
    HW3.accuracy(x, X_test, Y_test)
end
```

```julia
train_acc1 = map(hist1.x) do x
    HW3.accuracy(x, X_train, Y_train)
end

train_acc2 = map(hist2.x) do x
    HW3.accuracy(x, X_train, Y_train)
end;
```

[96]:

```julia
p1 = plot(train_acc1, ylim=(0.9, Inf), title="train accuracy", label="without l1 penalty", alpha=0.8)
plot!(p1, train_acc2, label="with l1 penalty", alpha=0.8)

p2 = plot(test_acc1, ylim=(0.9, Inf), title="test accuracy", label="", alpha=0.8)
plot!(p2, test_acc2, label="", alpha=0.8)

plot(p1, p2)
```

[96]:



Problem 3

March 17, 2023

[1]:

```
import cvxpy as cvx
import numpy as np
import imageio.v3 as iio
import scipy
import random
import matplotlib.pyplot as plt
```

[2]:

```
Y = iio.imread('SheppLogan_150x150.png')
Y = Y / Y.max()
n1, n2 = Y.shape
```

[3]:

```
orig_shape = Y.shape
flat = Y.flatten()
n_mutated = len(flat) // 10
rand_idxs = random.sample(range(0,len(flat)-1), n_mutated)
flat[rand_idxs] += np.random.rand(n_mutated).astype(np.float32)
Y_noisy = flat.reshape(orig_shape)
```

[4]:

```
D = np.zeros(Y.shape)
for i in range(n1-1):
    D[i,i] = -1
    D[i, i+1] = 1

D[n1-1, n2-1] = -1
Lh_tilde = scipy.sparse.kron(D, np.identity(n1))
Lv_tilde = scipy.sparse.kron(np.identity(n2), D)
```

[5]:

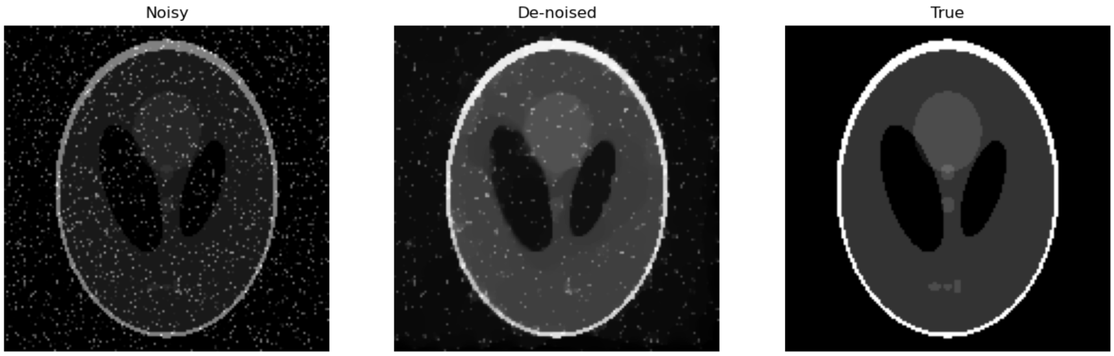
```
def TV(X):
    X = X.flatten()
    y_h = Lh_tilde @ X
    y_v = Lv_tilde @ X
    y = np.vstack((y_h, y_v))
    return np.sum(np.linalg.norm(y, 2, axis=0))

def TV_cvx(X):
    X = X.flatten()
    y_h = Lh_tilde @ X
    y_v = Lv_tilde @ X
    y = cvx.vstack((y_h, y_v))
    return cvx.sum(cvx.norm(y, 2, axis=0))

tau = 0.25*TV(Y_noisy)
```

[6]:

```
X = cvx.Variable((n1,n2))
objective = cvx.Minimize(0.5*cvx.norm(Y_noisy-X,'fro'))
constraints = [0 <= X, X <= 1, TV_cvx(X) <= tau]
prob = cvx.Problem(objective, constraints)
result = prob.solve()
Y_pred = X.value
```

[7]:

```
fig, axes = plt.subplots(1,3,figsize=(15,7))
axes[0].imshow(Y_noisy, cmap='gray')
axes[0].axis('off')
axes[0].set_title("Noisy")
axes[1].imshow(Y_pred, cmap='gray')
axes[1].axis('off')
axes[1].set_title("De-noised")
axes[2].imshow(Y, cmap='gray')
axes[2].axis('off')
axes[2].set_title("True")
plt.show()
```


Noisy
De-noised
True