q5

October 28, 2019

0.1 Question 4 (8 marks): implement accelerated gradient for the Total-Variation denoising problem. Use the pseudo-Huber function to smooth the problem. Use the Lipschitz constant that you obtained in Q3. Do not include computation of the Lipschitz constant in this question. You can do it in Q3 and the time for computing the Lipschitz constant will not be taken into account.

[ ]:
```python
def accelerated_gd_lip(x0, epsilon, lambda_, mu, max_iterations):
    z = x0.copy().toarray()
    x_updated = x0.copy().toarray()
    f_vals = []
    norm_vals = []
    t1 = time.time()
    alpha = 1.0 / L
    lamb = 1.0
    for i in range(1, max_iterations+1):
        gamma = 2.0 / i if i > 4 else 0
        y = (1.0 - gamma) * x_updated + gamma * z
        current_grad = grad_f_tv2(lamb=lambda_, mu=mu, x=y, z_n=noisy_image_vec)
        current_grad_norm = np.linalg.norm(current_grad)
        if current_grad_norm <= epsilon:
            break
        norm_vals.append(current_grad_norm)

        z = z - alpha * (gamma / lamb) * current_grad
        f_vals.append(f_tv(lamb=lambda_, mu=mu, x=x_updated,␣
↪z_n=noisy_image_vec))
        x_updated = y - alpha * current_grad
        lamb = (1.0 - gamma) * lamb
        f_diff = (f_vals[-1] - f_vals[-2]) if len(f_vals) > 1 else None
        grad_diff = (norm_vals[-1] - norm_vals[-2]) if len(norm_vals) > 1 else␣
↪None
        print(f"Step = {i}: alpha = {alpha}, gamma = {gamma}, lambda = {lamb},␣
↪Function = {f_vals[-1]}, Function Diff. =  {f_diff}, Grad. Norm =␣
↪{norm_vals[-1]}, Grad. Norm. Diff. = {grad_diff}")
    t2 = time.time()
    print(f"Iterations (Total) time = {t2-t1}")
    return x_updated, np.array(f_vals)
```

0.2 Call accelerated gradient to denoise the image. Use the same λ and μ that you used in Q1.

[ ]:
```python
# Initialize parameters of gradient descent
lambda_ = 45
mu = 0.1
epsilon = 1.0e-2
max_iterations = 200

# Set x0 equal to the vectorized noisy image.
# Write your code here.
optimized_gd_acc_lip, f_vals_acc_lip = accelerated_gd_lip(x0=noisy_image_vec,
                                                          lambda_=lambda_,
                                                          mu=mu,
```

```
                                                                           epsilon=epsilon,
                                                                          ␣
↪max_iterations=max_iterations)
```