JC Functions Tutorial

December 14, 2021

1 Function Demonstrations

[1]:
```
include("./JC_functions.jl")
```

[1]:
```
k_means
```

1.0.1 Gradient Descent

[2]:
```
function f(x)
    return x[1]^3 + x[2]^2
end

function  f(x)
    return [3. * x[1]^2; 2. * x[2]]
end

function  2f(x)
    return [6. * x[1]    0.;  0. 2.]
end

x = gradient_descent(f,  f, [1.; 2.])
println("Solution = $x")
g = norm( f(x))
println("Norm of gradient = $g")
```

```
Number of iterations: 1825400.
Solution = [0.00018257417735110227, 1.23e-321]
Norm of gradient = 9.999999070629523e-8
```

1.0.2 Newton's Method

[3]:
```
x = newton_method(f,  f,  2f, [1.; 2.])
println("Solution = $x")
g = norm( f(x))
println("Norm of gradient = $g")
```

```
Number of iterations: 13.
Solution = [0.0001220703125, 0.0]
Norm of gradient = 4.470348358154297e-8
```

1.0.3 Nonnegative Matrix Factorization

[4]:
```
X = [1 2 3;
     4 5 6;
     7 8 9;
     10 11 12]

W, H = nonnegative_matrix_factorization(X, 2)

display(W)
display(H)
display(W * H)
```

```
4×2 Array{Float64,2}:
 0.316025  0.316025
 0.771565  0.771565
 1.2271    1.2271
 1.68265   1.68265

2×3 Array{Float64,2}:
 2.86337  3.26054  3.65771
 2.86337  3.26054  3.65771

4×3 Array{Float64,2}:
 1.80979   2.06082   2.31185
 4.41855   5.03144   5.64432
 7.02731   8.00205   8.97679
 9.63607  10.9727   12.3093
```

1.0.4 AdaBoost

[5]:
```
n = 8
p = 4

X = [1. 1. 1. 1.;
     -1. -1. -1. 1.;
     -1. -1. -1. 1.;
     -1. -1. 1. 1.;
     1. 1. 1. 1.;
     1. 1. 1. 1.;
     1. -1. 1. 1.;
     1. -1. -1. 1.]

y = [1. 1. 1. 1. -1. -1. -1. -1.]

F = adaboost(X, y)

global correct = 0
for r in 1:n
```

```
    pred = sign(F(X[r, :]))
    if pred == y[r]
        global correct += 1
    end
end

train_acc = correct / n

println("Accuracy = $train_acc")
```

```
Number of iterations: 5.
Accuracy = 0.875
```

1.0.5 k-means Clustering

[6]:
```
X = [2. 2.;
     -1. -1.;
     -2. -2.;
     3. 3.;
     0. 0.;
     1. 0.;
     2. 1.;
     0. 1.]

centroids, nearest = k_means(X, 2)

println("Centroids: ")
println(centroids)

println()
println("Cluster-point pairs:")
println(nearest)
```

```
Centroids:
[2.333333333333333 1.9999999999999998; -0.4000000000000001 -0.4000000000000001]

Cluster-point pairs:
[1.0, 2.0, 2.0, 1.0, 2.0, 2.0, 1.0, 2.0]
```

[ ]:
```

```