OM Project Report

Kritika Agrawal (201502061)
Aakash Mittal (201501037)

Gradient Step

```
Initialize regularization parameters
eta = 1./(lamb * t)
stepDirection = {}

Initialize stepDirection with key corresponding to bias term

stepDirection[self.biasKey] = 0

Compute the subgradient of our loss function at the present update set
for obs in updateSet:
    if len(obs) < 2: continue

    y = int(obs[-1])
    assert (y==-1 or y == 1)

    Compute subgradient of bias(intercept) term
    stepDirection[self.biasKey] += y

     Compute subgradient of features
    features = obs[1:-1]
    for coord in features:
        featureName, featureVal = self.getVal(coord)
        featureVal *= y
        if featureName not in stepDirection:
            stepDirection[featureName] = 0.0
        stepDirection[featureName] += featureVal

scaling = eta/k

 Update weight coefficients
for i,(key,val) in enumerate(stepDirection.items()):
    indx = self._hash(key)
    rad = -1

    self.weights[indx] *= (1 - (eta * lamb))
    self.weights[indx] += scaling * val * rad

return stepDirection.keys()
```

Project Step

Calculate the norm in squared form

```
normSquared = 0.

for feature in stepKeys:
    indx = self._hash(feature)
    normSquared += self.weights[indx] ** 2
```

Calculate the scaling factor based on norm and lambda

```
scaling = 1
if normSquared != 0:
    scaling = min(1, (1/(lamb ** .5))/(normSquared ** .5))
```

Update the weights

```
if scaling != 1:
    for feature in stepKeys:
        indx = self._hash(feature)
        self.weights[indx] *= scaling
```

Kernel Used : None

Dataset Used : Breast cancer dataset

Accuracy achieved : 0.8759124087591241

Time take : 0.044s required for training on 545 samples with 9 features each and testing on 136 test samples.