**Subgradient decent for kernel multiclass SVM:**

Algorithm:

Input:

* T – number of iterations
* C – regularization constant
* eta – learning rate
* train\_dataset – training data set (includes both the data and labels)
* K – set of labels

Init:

Define an SVM object -   
 SVM: {

misclassified\_count = 0,

misclassified\_samples = []

misclassified\_coeffs = []

}  
 foreach i in k:

svm\_list[i] = new SVM object

Algo:

1. for t = 1..T
   1. eta\_step 🡨 eta / i
   2. x, y 🡨 random sample from train\_dataset
   3. calc prediction for each class (svm\_predict is a list of w\*x):
   4. svm\_predict = []

foreach i in K:

svm\_predict[i] 🡨 ***predict(*svm\_list[i], x)**

* 1. js 🡨 argmaxq(svm\_predict[q] – svm\_predict[y]) (q from K-{y})
  2. delta 🡨 svm\_predict[j] – svm\_predict[y]
  3. if delta < 1:

# update wj

*foreach j in js:* ***update \_svm(list\_svm[j], -x, eta\_step, C)***

# update wy

***update\_svm(list\_svm[y], +x, eta\_step, C)***

*# update the others*

for i=K-{y} – js:

***increment\_svm(list\_svm[y], none, eta\_step, C)***

#calc w\*sample

**predict(svm, sample):**

1. res 🡨 0
2. for i = 0..svm.misclassified\_count:

res += svm.misclassified\_coeffs[i] \* kernel (misclassified\_samples[i], sample)

**increment\_svm(svm, misclassified\_sample, eta\_step, C):**

1. svm.misclassified\_coeff \*= (1-eta\_step)
2. if (sample == none) return
3. svm.misclassified\_samples .append(misclassified\_sample)
4. svm.misclassified\_coeff .append(eta\_step \* C)
5. svm.misclassified\_count++