Regularized Spectral Learning (RSL) Toolbox

Susan Shortreed
susanms@stat.washington.edu

Marina Meila
mmp@stat.washington.edu

September 27, 2005

1 Introduction

The RSL toolbox contains the code to run the spectral learning algorithm presented in 'Regularized Spectral Learning' AISTATS 2005, Meila, M., Shortreed, S., Liang, X. The toolbox contains the script `RSL.m`, which runs the learning algorithm, as well as all the functions necessary to run this script. The main directory, **RSL_library** contains `install_RSL.m`, the code which adds all the library directories to the Matlab session. There are two subdirectories **ClustDep** and **ClustInd** which contain code for learning cluster dependent parameters and cluster independent parameters respectively. The cluster independent parameters make the similarity function of the form:

$$S_{ij} = \exp\left(-\sum_{f=1}^{\text{nf}} \theta_f x_{ij,f}\right)$$

Where θ_f is the parameter associated with the f^{th} feature, for all points. The cluster dependent parameters make the similarity function of the form:

$$S_{ij} = \exp\left(-\sum_{f=1}^{\text{nf}} \theta_{i,f}\theta_{j,f} x_{ij,f}\right)$$

Where $\theta_{i,f}$ is the parameter associated with the f^{th} feature and the cluster to which point i belongs and $\theta_{j,f}$ is the parameter associated with the f^{th} feature and the cluster to which point j belongs

2 Using the Toolbox

2.1 Starting the Toolbox and Example Code

To install the Regularized Spectral Learning, RSL, toolbox, set RSL_HOME to a string containing the name of the directory in which the toolbox was saved and make sure that install_RSL is in a director which matlab can access. The RSL toolbox uses functions from the Spectral toolbox so it must also be installed. To obtain is go to: `www.ms.washington.edu/~spectral/` If it is already installed in the current matlab session nothing needs to be done. If it has not been installed then SPECTRAL_HOME should be set to a string containing the name of the directory it has been saved. With this information both libraries will be installed by running the small script install_RSL. An example of how to use the learning code, the script `RSL.m`, is contained in `example.m`.

2.2 Input / Output

The objective of `RSL.m` is to learn a set of parameters which will optimize the function, J = gap - α*(eigengap^2-ϕ). For more information about this approach see [1]. A general line search method, following the Armijo rule, is used. In order for `RSL.m` to run, several variables need to be set as global variables: `ff vtrue S vv ll`. The information below is a description of the inputs require for the learning code and the output it produces, it is also contained in the comments of `RSL_CI.m`, cluster independent parameters, `RSL_CD` cluster dependent parameters.

2.2.1 Input

- `k` = number of clusters
- `n` = number of data points
- `nf` = number of features
- `ff( nf, n, n )` = array of dissimilarity feature matrices, MUST BE >= 0 !!
- `niter` = maximum number of iterations for learning the parameters
- `iniweight` = initial parameter values, dimensions are different in cluster independent and cluster dependent weights
- `alpha` = regularizing parameter on the eigengap

- `phi` = the value the square of eigengap should be larger than
- `assignment_true` = assignment vector with the true clustering, needs to have values between 1 and k

Everything below here should be set in the parameter file.

- `Flag = struct('sqeg',1,'saveGrad',0,...)`
  - `Flag.clustInd` - if true cluster independent weights are learned, if false cluster dependent
    Note: the size of the initial weight must match up to the type of weights learned.
    cluster independent weight: 1 x n; cluster dependent weights: k x n.
  - `Flag.sqeg` - if true J contains the square of eigengap, if false just the eigengap
  - `Flag.saveGrad` - if true will return the gradient values at each step in `dJdw_save`
  - `Flag.saveEigs` - if true saves the first (k+1) eigenvalues at each step in `ll_save`
  - `Flag.useL` - if true uses $L = D^{-1/2}S^{-1/2}$ to cluster, if false uses $P = D^{-1}S$
  - `Flag.CE_save` - if true then save the clustering error at every learning iteration in`ce_save`
  - `Flag.lowmem` - if true ff will be saved on disk (ffTemp.mat) and read only when needed.
- `global KMEANS_THRESHOLD` = threshold for kmeans clustering of eigenvectors
- `global KMEANS_MAX_ITER` = max number of iterations for kmeans
- `smax` = maximum number of optimizing steps in line search
- `Gammas` = first and largest learning step size in line search
- `Gamma` = how much reduce step size
- `beta` = a value of parameters is said to reduce J if
  J(weight - grad*gammas) < J(weight) - beta*gammas

2.2.2 Output

- `ncut_save` contains the multiway normalized cut at each iteration
- `gap_save` contains the value of the gap at each step
- `J_save` contains the value of J at each step
- `eigengap_save` contains the value of the eigengap at each step
- `weight_save` contains the value of the parameters at each step
- `step_save` contains the size of the learning step taken at each iteration
- `timeElapsed` number of seconds `rsl.m` script took to run.

OPTIONAL OUTPUT: These objects will be returned if the specific flags are set to true.

- if `Flag.saveGrad`, `dJdw_save` contains the value of the gradient for each step
- if `Flag.saveEigs`, `ll_save` contains the largest k+1 eigenvalues at each step
- if `Flag.CE_save`, `ce_save` contains the clustering error of the estimated clustering if the parameters at each iteration were used to make a similarity matrix.

3 References

[1] Meila, M., Shortreed, S. and Liang, X. "Regularized Spectral Learning". AISTATS 2005.

[2] Shortreed, S. and Meila, M. "Unsupervised Spectral Learning", UAI 2005