

CS 228, Winter 2009

Programming Assignment #3—Learning in TAS

Due: Saturday March 14, 2009 @ 11:59 pm

1 Introduction

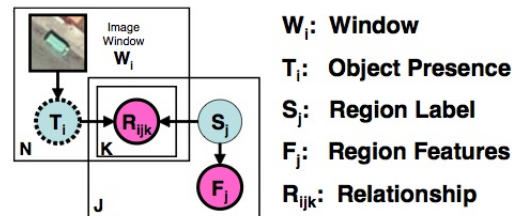
In this programming assignment, you will revisit the TAS model from PA1. In PA1 we used the TAS model to improve our identification of cars in images, using the context of the image. While the identification of the cars improved, there were several shortcomings of the model we used in PA1. One of these, which was alluded to by one of the PA1 questions and which several of you noticed, was that the inference process didn't help with the labeling of the segments. This problem arose in part because the parameters of the model were hand-coded. In this assignment, you will instead learn the parameters of the model from data, both labeled and unlabeled, and then see how well this performs on some test images. Once you have completed this, you will implement an extension to TAS of your own choosing, and report on the effectiveness of your implementation of this extension. This assignment is due at **11:59pm on Saturday March 14, 2009**.

2 [50 points] Basic Part

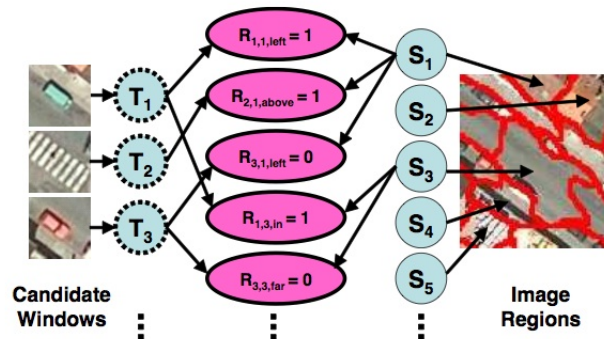
In this portion of the assignment, you will learn the parameters of the TAS plate model from different types of data. In all the training tasks, we will only learn the parameters of relation variable R , the segmentation feature variable F , and the segmentation label variable S . We will not learn the prior for the detection indicator T , since it is given from the output of the generic object detection algorithm.

2.1 TAS Model

In this programming assignment the TAS model includes region features F , in addition to the region labels S , object detections T and relationships R that were used in PA1. The provided data is a set of images. Each image has been processed by a generic object recognizer, generating the detections. Each image has also been (hand)-segmented into 3 categories (1 = road, 2 = tree, 3 = roof). In addition, each segment has a single multinomial feature (F), which corresponds roughly to the average color of the segment. The TAS model will make use of all of this information to attempt to identify cars in the images. You have been provided a training set of 15 images and a test set of 5 images, with which the different parts of this assignment will work.



(a) TAS plate model



(b) TAS ground network

2.2 [30 points] Code

All code for this assignment should be copied from

</afs/ir/class/cs228/ProgrammingAssignments/PA3/assignment/>.

You should be able to access this by logging into the Stanford Unix machines (e.g., cardinal.stanford.edu) using your SUNet ID. You can also use a copy of Matlab on any of the machines that allow CPU-intensive jobs (tree, vine, bramble, hedge).

Your job is to fill in the missing portions of code – marked with **YOUR CODE HERE** – in the files described below. The script **TestPA3.m** is the main routine that sets up the experiments to be run and also evaluates whether your implementation is correct. Initially, running this script without making any code changes should result in all tests failing (though they should run without error).

The experiments run in **TestPA3.m** and the function that must be implemented for each one are enumerated below. For each experiment, sample inputs and outputs are loaded, and the test passes if the output produced by your implementation matches the correct outputs that are loaded in. Note that when you submit your final implementation it will be tested with different inputs and outputs.

Once the tests confirm that your function implementations are giving correct outputs, you can uncomment the code at the bottom of **TestPA3.m** which will run your code on a training set and test it on a test set, allowing you to see the performance of the learned models on the five test images.

2.2.1 [10 points] Learning with fully observed data

In this portion we assume that we are given as data a set of images. Each image has been segmented, and each segment has had its average color observed and been correctly labeled. We also have labeled detections (places in the image where the car detector found a car). A detection is labeled as either a true car (correct detection), or not a car (wrong detection). Our task is to use this fully labeled data to learn parameters for the TAS model.

1. [5 points] **GetTasSuffStats.m**: This function should collect the sufficient statistics for the TAS model given a training set. We use Laplace smoothing to avoid 0 statistics.
2. [5 points] **EstimateTasParams.m**: This function should estimate the TAS model parameters given the (expected) sufficient statistics.

You should be able to learn a model from the fully observed training set.

2.2.2 [20 points] Learning with partially observed data

There are several different ways that our data could be partially observed. In this section we will explore two of those ways. We will assume throughout that each region feature (F) is observed.

- Case 1: The labels for all detections are observed. In this case, we assume that we don't have the labels for any of the segments, but that we know the labels of all the detections.
 1. [7 points] **GetTasExpSuffStats.m**: This function will calculate the expected sufficient statistics for the model. It assumes that all detections are observed. You are given a helper function called **LogPosteriorSeglabel.m** which will compute the log posterior of all the segmentation variable in the network. (Hint: write down the posterior for the family of a relation variable and the family of a feature variable to see how can you compute it with the help of the posterior of the segmentation variable.)
 2. [6 points] **LearnTasModelEM.m**: This function uses EM to learn the model parameters for TAS. It calls **GetTasExpSuffStats.m** and **EstimateTasParams.m** to achieve this.
- Case 2: Some of the detection labels are unobserved. In this case we again assume that we don't have the labels for any of the segments, but additionally we assume that we do not know the label of all the detections. In this case, the **LearnTasModelEM.m** function will have only one modification. It will call the following function **GetTasExpSuffStats1.m** instead of **GetTasExpSuffStats.m**. This will enable you to run EM without assuming that having all of the detections observed.
 1. [7 points] **GetTasExpSuffStats1.m**: This function will calculate the expected sufficient statistics for the model, without assuming that the detections were observed. Since running exact/approximate inference on the TAS model is slow, you are provided a fake approximate inference function called **ApproxInferenceFake.m** which will load and return the calibrated marginals and cluster potentials pre-calculated from the cluster graph corresponding to the network. **ApproxInferenceFake.m** is interchangeable with **ApproximateInference.m**. The reason we provide it here is because inference takes a long time, and so the fake function will save you time as you test out your implementation.

2.3 [20 points] Questions

Some of the questions are qualitative in nature, and their purpose is to get you to think about the properties of these algorithms. A few sentences are sufficient for each question, and be specific about any results you cite (e.g., the segmentation labels in 3 out of 4 test instances improved). In cases where you want to view the performance of a set of model parameters on the test data, you can use `TestTAS.m` (see Details section for details).

1. **[3 points]** In the EM code provided, the model parameters are randomly initialized. Consider the case where we set all of the CPD's in the model to be the uniform distribution. Modify the code to implement this change (Do this for the case when the segment labels are unobserved but the detections are fully observed). Describe in a few sentences what happens to EM's performance in this case.
2. **[3 points]** One parameter that you provided to the EM algorithm was the cardinality of the hidden segment label variables, which in this case was 3. Modify your code so that EM assumes that the hidden segment variables have only 2 different possible labels. Run this code, assuming that none of the segment labels are observed, but that all of the detections are observed (along with the segment features). Repeat again with 4 possible labels. Compare the resulting performance of these models on the test data (by examining and comparing the ROC curves, say). In a few sentences, describe these results. Describe briefly how you would go about determining the best cardinality to assign a hidden variable, given no previous information (as opposed to this case where we knew that the segmentations were created with specifically 3 labels.)
3. **[3 points]** In considering partially observed data, we made a distinction between whether or not all of the detections are observed. Referencing the TAS plate model, explain briefly what difference this assumption makes to the complexity of our learning task.
4. **[4 points]** As mentioned earlier, the PA1 TAS model failed to help with segmentation. Train a model on the fully observed training set and run it on the test set. How do the posteriors over segmentation labels compare to the learned priors? Which label is this model able to most accurately infer? What is the posterior over segmentation labels if we only observe the segmentation color feature F ? (Provide your answer in table form, where for each possible observation F you provide the associated posterior over segmentation labels.) As a reminder, the segmentations used on this assignment 1 = road, 2 = tree, and 3 = roof.
5. **[4 points]** In this question we would like you to compare the performance of the 2 models (fully observed data, unobserved segment labels and fully observed detection labels, we won't have you use the model resulting from running fake inference). Run each of these 2 models on the test images provided and view and compare their ROC curves. Which method produced the best results (explain in a sentence or two why you think this)? Briefly describe the conditions (what types of training data, quantity of training data, etc) under which you would elect to use each of these methods for specifying the parameters of a model.
6. **[3 points]** To begin EM, we used a random assignment to the parameters of the model. Modify the code (shouldn't require hardly any work), to do multiple runs of EM with different random starting points. (Do this for the fully observed detections case). Do the different runs of EM converge to the same parameters? To the same log-likelihood? Do they converge at the same pace in terms of the number of EM iterations?

You must include the answers to these questions in an ASCII text file called README when you submit your code.

Once you have completed everything up to this point, and answered the questions, you can submit this assignment in the usual manner. The submission for the extension part of PA3 will be a separate .pdf, emailed to the TAs.

2.4 Details

The files **TAS_DATA_TRAIN.mat** and **TAS_DATA_TEST.mat** each contain an array of data for the TAS network. Each of them has the following fields:

- **image_data** which has the following fields (Note that we have additional fields compared to PA1):
 - **dets**: Each row has the x and y coordinates of two corners of the bounding box of a single candidate detection
 - **scores**: Has the scores corresponding to each candidate detection, as given by the object detector
 - **gt**: The true detections
 - **image_filename**: The name of the corresponding image
 - **segindices**: A matrix specifying which segment each image pixel belongs to
 - **seglabels**: A matrix specifying the label of the segment of each pixel as assigned by the segmenter
 - **rels**: $\text{rels}(t, s)$ represents the relation observed between detection window t , and segmentation s .
 - **gt_dets_indicator**: $\text{gt_dets_indicator}(d)$ the ground truth indicator for each detection window. $\text{gt_dets_indicator}(t) = 1$ means that there is no car in the t -th detection window, and $\text{gt_dets_indicator}(d) = 2$ means that there is indeed a car in the d -th detection window.
 - **seghues**: $\text{seghues}(i)$ is the feature value for the i -th segmentation.
- **TAS_model** which has the following fields:
 - **TAS_network**: The network for the image. In addition to the edges, names and dim fields, it also has num_segs (number of segmentations) and num_dets (number of detections) fields.
 - **TAS_factors**: TAS factors for the network.
 - **TAS_cluster_graph**: TAS cluster graph for the network.
 - **TAS_EVIDENCE**: Evidence for detection ground truth indicator, relations and segmentation features.

In addition, the following two types of data are used throughout this programming assignment.

- **sufficient statistics (ss)**: This accumulates the sufficient statistics necessary to do the learning. It was the following fields:
 - **seg_label_counts**: This keeps the counter for each of the segment labels.
 - **rels_counts**: This keeps the counters for the combinations of R , S and T .

- `feature_counts`: This keeps the counters for the combinations of F and S .
- `tas_params`: This keeps track of parameters of the learning process including:
 - `RK`: The number of different relations in the model.
 - `LK`: The number of different segment labels.
 - `TK`: The number of different detection labels.
 - `FK`: The number of different feature values.
 - `EM_RESTARTS`: The number of times the EM portion should restart at a random location.

The following files are provided for your use:

- **Initialize.m** This function is used to set the learning parameters `tas_params` as mentioned above.
- **LearnTasModel.m** This function learns the TAS parameters with fully observed dataset.
- **LogPosteriorSeglabel.m** This function computes the log of the posterior over each S with all the segmentation features, relations and detection indicators observed.
- **RandomTasParams.m** This function randomly generates a valid set of parameters for the TAS model. Used to randomly start EM.
- **TasDataLogLikelihood.m** This function computes the log likelihood of the training dataset given the model parameters.
- **UpdateTASModel.m** This function updates CPDs of each factor for a TAS network using the specified model params. This function can be used to update our model using the learned parameters.
- **TestTAS.m** This function takes in an array of test data and a set of model parameters, and evaluates the model parameters on the test data. Very useful for comparing different models.

3 [50 points] Extension Part

3.1 Introduction

You have now completed two programming assignments dealing with the TAS model. The model and methods used in the programming assignments have many areas that could be improved upon. For example, so far we have used hand-labeled segmentations, at varying degrees of realism, but have not yet used the output of an actual segmenter.

The second part of this programming assignment is an exercise in real-world use of the methods covered in this class. This is an open-ended assignment which gives you a chance to apply what you have learned this quarter.

3.2 Assignment

Your task is as follows. Take the TAS model as it is in the first half of this assignment and extend it in some direction with the goal of improving its performance. You could consider this as if you work for a company, and your boss gives you the TAS model used here and says, “Make it better.”. For this assignment you will be expected to select at least one area in which you can extend the TAS model and implement an extension in this area. You will be evaluated not on your results, but on your techniques and evaluation of those techniques. Obviously, one cannot always guarantee that the first idea one has to improve a system will work, but regardless of whether or not performance improves, rigorous analysis of this can be performed to communicate convincingly to others the effectiveness of the idea. In the following section we provide a list of possible extensions, although you are not limited to items on the list. Please feel free to contact the TAs during office hours or at (cs228-qa@cs.stanford.edu) for suggestions or approval of your extension ideas, even if you choose an extension on the list.

You will submit a report of your extension, which is what your grade will be based on for this portion of the programming assignment. Your report should include

- [$\frac{1}{2}$ to 1 page] A description of the method/extension you implemented, including a sufficiently clear explanation to convince the reader that it was implemented correctly.
- [3 to 4 pages] Analysis of the method/extension implemented. Such analysis should include a discussion of the analysis methods used, actual analysis of the results of the method, as well as any charts or graphs necessary to make the desired points.

3.3 Analysis

As mentioned earlier, you will not be graded on whether or not your extension actually improves the performance of the TAS system on the task of identifying cars in images, but rather on the scope and correctness of your implementation and the appropriateness of your analysis. To help guide you in this regard, we provide a few examples what kinds of things would be involved in good analysis.

- “This graph shows the results from 20 random restarts, and indicates that we have a problem getting stuck in local minima”
- “Here is a comparison of three minor variants of our algorithm, each with slightly different parameters settings”

Additionally, you should include analysis of intermediate results as well as your final results.

- “These are the features we selected in the first stage, and this is how we chose them”
- “This plot shows the likelihood in each iteration, and we can see that it converges within just a few iterations”

3.4 Possible Extensions

1. Incorporating features of the segmentation

In the current model, the only feature of each segmentation region that is used is the color of each segment. This could be extended to incorporate other features of the each region (size, shape, etc), which could help identify unlabeled segments.

2. Including edges between the segmentations
As mentioned in the lecture, image segmentation MRFs typically include edge potentials in the pairwise Markov network for an image. Similarly, we may add edges between adjacent segmentations.
3. Distances between detections
It is clear from looking at the images that true detections seem to be close together, since cars typically appear on roads. You can extend the TAS model to take into account the distance a detection is from other detections, much like the relation variable currently used between segments and detections, and use this to help find cars.
4. Using other cluster graphs or clique trees
The exact inference or approximate inference take a relatively long time to run on TAS model given the current clique trees or cluster graphs. What if we use different clique trees or cluster graphs (different scheduling)?
5. Using sampling-based inference
Instead of BP, another class of inference techniques is sampling-based methods. You can implement a sampling-based inference method evaluate its performance and compare it to other inference methods.
6. Structure learning (learning the active relations)
Instead of only having 3 relations in our simplified model, we can have a richer set of the relations. For example, we can have relations based on the directions. Moreover, in general you don't know what are the best set of relations in advance, so it would be better to define an overcomplete set of relations and learn which are the effective ones. This can be cast as a structure learning problem, where each relation is modeled as a binary random variable indicating whether this relation is true or not, and we can use structure learning to find out which ones should be active (in this case, detection and segmentation variables will have edges connected to it) and which ones should be inactive (in this case, no edges are connected to it).
7. Finding roads
Instead of segmentation, you could apply a road-finding algorithm to an image, which would have the purpose of identifying the road in the image. This could be done by fitting lines to the image or using another method of your choosing. The information about the road's location would then be incorporated into the model.

If you have ideas for extensions outside of this list, that is perfectly fine, and even encouraged, we simply urge you to discuss it with the TAs. We can help ensure that the scope of your extension is appropriate. Even if you have any questions about guidance and confirmation for things on the list, don't hesitate to discuss it with the TAs (cs228-qa@cs.stanford.edu).

Your submission for this section will be separate from the first portion of this assignment, and should be in the form of a .pdf document, emailed to (cs228-qa@cs.stanford.edu) before the deadline.

4 Details

All of the functions from PA1 and the previous part of PA3 are still available for your use. In addition, the following supporting functions or data have been provided for you to use.

- **CreateTASModel.m** This function takes an `image_data` object, along with a `tas_params` object, and returns a `TAS_model`, complete with factors, network and cluster graph.
- **CreateTASCliqueTree.m** This function takes in a network and set of factors and generates a corresponding clique tree. This is used by **CreateTASModel.m**
- **CreateTASClusterGraph.m** This function takes in a network and set of factors and generates a corresponding cluster graph. This is used by **CreateTASModel.m**
- **CreateTASDataSet.m** This function loads in image data and creates the TAS model for it. This function can be used to prepare training and test data instances.
- **UpdateGroundTruthDetIndicator.m** This function updates the `image_data` by adding the ground truth of the detection indicator.
- **UpdateRelationships.m** This function updates the `image_data` by adding the `rels` field using the observed relations between each pair of detection and segmentation.
- **Original TAS data** `/assignment/TAS/` contains the full data from the original TAS experiments. These include images, full number of detections with true labels as well as fully and automatically segmented data. This is a great resource for any extensions which seek to extend the ability of our TAS model to deal with a larger number of detections and segments.

If you are attempting an extension of great scope, and feel that some low level function or utility should exist but isn't currently provided, please contact the TAs, and we might be able to help provide this functionality.