CS 696i: Computer Vision Section

Computational Assignment

Due April 5

This assignment should be done as a group.

The mathematical notation and translation into a program may prove a challenging for those not in ECE or CS. However, it is your chance to learn from very skilled personal tutors! For those that take a lead role due to the technical nature of the assignment, I will be soliciting opinions from your group as to how well you helped others understand what is going on.

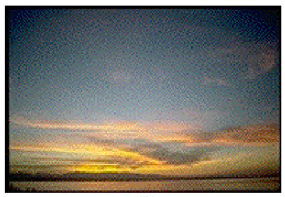
In this assignment you will implement a simple statistical model that predicts words from image regions based on images with associated keywords (see Figure 1). You will notice that we do not know which part of the image each word comes from. Nonetheless we can make progress due to two observations:

1. If the images with a specific word like "sky" were considered together, their regions would on average be more like "sky" and less like the global average. Thus we could estimate what the visual features for "sky" might be.
2. If we had an idea what the visual features for "sky" are, then, we can assign "sky" to those regions, and thereby ignore them when we try to estimated the visual features for a different word (e.g., "tiger"), thereby making step 1. more accurate.

It may be helpful to consider an analogy with statistical machine translation (see Figure 2).

Figure 1.

TIGER CAT WATER GRASS



1090
SUN CLOUDS
WATER SKY

143078
MOUNTAINS TREES
aspens VALLEY

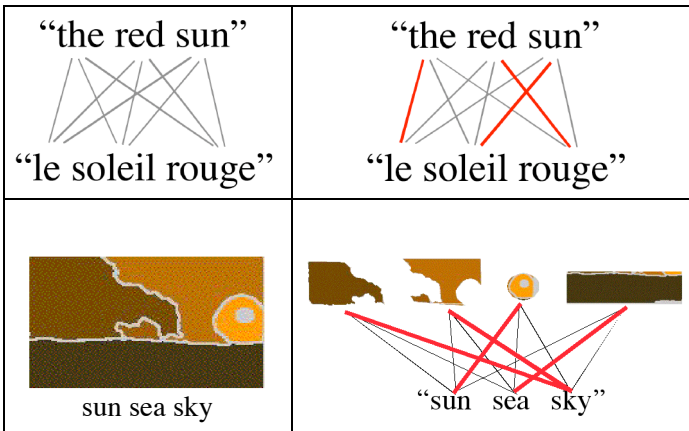


Figure 2. The analogy of statistical machine translation and computer vision. The top left box shows a simple English phrase and its French translation. The correspondences are not known a priori. The top right box shows the correspondences which can be obtained given additional sentences with words which overlap these ones, such as "the red car". The bottom row shows the same process applied to image pieces. In this case, more than one image piece maps to the same word. This occurs in language also, as does the case of some regions (English words) which do not map to any words (French words).

You will implement two programs. The first learns a model for the data and saves the corresponding Matlab matrices to disk. The second reads in the learned model, and processes new images to label their regions.

To greatly simplify the assignment, you will **not** need deal with images. The images have already been broken into regions, and the 10 largest regions have been associated with feature vectors which capture qualities like color and texture. Thus, N images will be represented by a matrix with 10*N rows. Each row item is a "feature".

There are 42 words, represented by a number from 1 to 42. The vocabulary will be provided, but you do not need to use it directly. Instead, you will work mostly with a matrix which gives the words that occur for each image

I will provide a method to put labels on the images so that you can see the results of your labors.

The infrastructure

The file http://kobus.ca/teaching/cs696i/spring05/ua_only/assignments/cs696i_a2.tar.gz (and the. zip version, cs696i_a2.zip, which Windows users may find more convenient) contains the following essential files;

0_continuous_item_mp	The feature vectors for 10*N regions for N images used for training
2_continuous_item_mp	The feature vectors for 10*H regions for H held out images used for testing
0_discrete_item_sparse_mp	The words associated with each of N images. Each image has a row of 4 integers from 1 to 42, or –99 which is used to deal with images which have fewer than 4 words.

as well as the following non-essential files that may be helpful for debugging, checking, or extending things.

words	The 42 words
2_discrete_item_sparse_mp	The words for the held out images which can be used to measure how well you are doing.
0_image_nums AND 2_image_nums	Numbers identifying the images.
con_feature_labels	A very terse record of the features used

Finally, the page: http://kobus.ca/teaching/cs696i/spring05/ua_only/assignments/label_images.html can be used to label images. To use it submit a file of integers, one row per each image with the following format:

data_set_id image_index word_num word_num ....

data_set_id is either 0 for training images or 2 for testing images and the image_index is the number from 1 to N or 1 to H of the image. These two numbers are followed by 0 to 10 integers between 1 and 42 which are assumed to be a word for the region in the order that they occur in the data files (largest to smallest). Any data which does not conform to this specification will be ignored without comment. I suggest testing this interface out ASAP with random input: For example:

2 100 1 2 3 4 5 6 7 8 9 10

The Model.

We will assume that there exists M latent (hidden) concepts indexed by the variable, *c*, with relative frequency of occurrence *P(c)*. This is the "prior" over concepts.

M is the main free variable that you have in this assignment. For this data, a value of M of about 100 or 200 seems reasonable to try once the code appears to work.

Each concept can statistically generate both words and regions. Words are generated by a simple frequency table. We denote this table by: *P(w|c)*, where *w* is the word, and we are conditioning on the concept, *c*. We will shortly specify a similar probability, $P(\mathbf{x} \mid c)$, which tells us how likely a feature vector, **x**, for a region comes from a given concept.

It is the goal to link feature vectors to words. Clearly, words and feature vectors are not statistically independent (otherwise our task would be hopeless!). However, a critical simplification, built into the model, is that the words and feature vectors are independent **given the concept.** This conditional independence is expressed as:

$$P(w,\mathbf{x} \mid c) = P(w \mid c)P(\mathbf{x} \mid c)$$

(Recall that if words and feature vectors were independent, we would write this as $P(w,\mathbf{x}) = P(w)P(\mathbf{x})$).

Region features are generated by Gaussian distribution with independent dimensions (i.e. the covariance matrix is diagonal). Thus every concept, *c*, is associated with a vector of means (one for each feature) and a vector of variances. The analogy to looking up probabilities in the table *P(w|c)* is a computation with Gaussians. To compute the contribution of one feature, i, we would use the familiar Gaussian distribution:

$$P(x_i \mid c) = \frac{1}{\sqrt{2\pi}\,\sigma(c)_i} \exp\left[-\frac{1}{2}\frac{(x_i - u(c)_i)^2}{\sigma(c)_i^2}\right]$$

where x_i is the i'th feature of the region under consideration.

To do all the features, 1 through *d*, we simply multiply the *d* probabilities together:

$$P(\mathbf{x} \mid c) = \prod_{i=1}^{d} P(x_i \mid c)$$

This is more commonly computed by:

$$P(\mathbf{x} \mid c) \quad \frac{1}{\left(\prod_{i=1}^{d} \sigma(c)_i\right)} \exp\left[-\frac{1}{2}\sum_{i=1}^{d} \frac{(x_i - u(c)_i)^2}{\sigma(c)_i^2}\right]$$

Here I have dropped the $(2\pi)^{d/2}$ as you can normally forget about it, but to be precise I changed the equal sign to proportionality sign, $\propto$. I will be using this sign to indicate a quantity which needs to be "normalized" before it is used as a probability. Normalization here means that the probabilities of all the possibilities need to sum to one---more on this below.

Concrete example: An "orange tiger" concept will have a much bigger number in the frequency table for "tiger" compared with "sky", and will have Gaussians for the features which favor orange stripy regions. A concept devoted to blue sky will have a larger $P(c)$ than one devoted to "cougar".

Statistical Inference: Given the model, how do we produce words for regions, and images?

We want the probability that a word, w, and a feature vector, **x**, come from the same concept. We need to consider all possibilities, namely that they come from any of the M concepts, weighted by how likely that possibility (concept) is. Recalling the conditional independence above:

$$P(w,\mathbf{x}) = \sum_c P(w \mid c)P(\mathbf{x} \mid c)P(c)$$

Technically, what we really want is P(w|**x**), so we might have to normalize:

$$P(w \mid \mathbf{x}) = \frac{P(w,\mathbf{x})}{\sum_w P(w,\mathbf{x})}$$

In this assignment, you simple need to find the word, w, which gives the max $P(w \mid \mathbf{x})$, so normalization is superfluous. However, it is important to realize that we have computed much more information than the maximally possible word---we have the entire distribution over all words for each region. The distribution can be used to defer on words which we are not sure (the max is not very close to one), or to label regions with additional choices (but this makes for messy figures).

To get the probability of a word for the image taken as whole, we can simply add the above expression for each region and renormalize (there are other ways---there is a complex issue underneath here). For this assignment, you don't need to produce words for images, but it is implicit in the computation of the log likelihood which is a key tool to verifying that the learning algorithm is implemented correctly:

Log likelihood: The log likelihood is proportional to the log of the probability that one sees all the data under the current model parameters. The algorithm adjusts the likelihood so that it increases on every iteration. Thus if this is not happening, there is an error. The likelihood of the data is the product of the likelihood of each and every image (i.i.d).

The likelihood of a **single** image and its words, using capital letters for sets of items, this quantity is:

$$P(W,X) = \prod_{w \in image} P(w \mid X) \prod_{\mathbf{x} \in image} P(\mathbf{x})$$

where $P(\mathbf{x}) = \sum_c P(\mathbf{x} \mid c)P(c)$ (Referred to later as ***)

and $P(w \mid X) = \sum_c P(w \mid c)P(c \mid X)$

where $P(c \mid X) \propto \prod_{\mathbf{x} \in image} P(c \mid \mathbf{x})$ (Best to re-normalize this one).

(The slight asymmetry is due to the fact that the model is translating from regions to words).

Intuitively you need to compute the product of P(W,X) for all images. We instead compute the log of this quantity:

$$LL = \prod_{images} \left[\prod_{w \in image} \log(P(w \mid X)) + \prod_{\mathbf{x} \in image} \log(P(x)) \right]$$

It says how likely the data you have is, given your model. The bigger the number, the better your model for that data. In this assignment we will tweak the model so that this quantity goes up at every stage.

Thus, if the LL does not go up, you have a bug. Note that as a quantity increases, so does its log. It is critical that you do this to ensure that the code is likely correct. (A report of the LL is also a deliverable).

Finally we can talk about learning!

To learn the model, we will use an algorithm called Expectation Maximization (EM) which is one of the most common algorithms in computer vision. By "learning the model", I mean estimating the following quantities:

The concept prior, P(c)
The word frequency table for each concept P(w|c)
The vector of means and variances for each concept, u(c) and v(c)

The key observation is the following chicken and egg situation:

> **Chicken**. If we had an estimate which concept the observed regions and words came from, then we could compute the above (this is the M step).
>
> **Egg**. If we knew the above, we could compute the expected value that the concepts are responsible for the observed regions and words (this is the E step).

The problem is that we know neither. We solve the problem by guessing an answer to the egg, and iteratively computing the chicken, then the egg, then the chicken .... (thinking about random genetic mutations should make it clear why I think the egg comes first).

Data structures:

In addition to the model, you will want to store the expectation that a given concept is responsible for a given word in a given document, P(c|w,i) and similarly for the regions: P(c|x,i). The size of the first is:

num_images*num_words_per_image* num_concepts

and the size of the second is:

num_images *num_regions_per_image *num_concepts.

Note that P(c|w,i) is limited to the words in each image, not the entire vocabulary. You will need to take care that rows containing missing words (the –99's in the data) are ignored.

The algorithm

(Recall that means that you have to scale the quantities by their sum so that they add up to one):

Initialization:

Guess P(c|w,i) and P(c|x,i). I suggest trying rand() raised to the power 5 and renormalizing. A simple uniform distribution seems on average to be too homogeneous. (Initialization is an art).

Loop:

M-Step:

$$P(c) \propto \sum_{images} \sum_{w} P(c \mid w,i) + \sum_{\mathbf{x}} P(c \mid \mathbf{x},i)$$

You need to normalize the RHS so that it sums to one over concepts, c, to get P(c).

$$P(w \mid c) \quad \sum_i P(c \mid w,i)$$

You need to normalize the RHS so that it sums to one over words, w, for every c, to get P(w|c))

$$\mathbf{u}(c) = \frac{\sum_{i,\mathbf{x} \in i} \mathbf{x} P(c \mid \mathbf{x},i)}{\sum_{i,\mathbf{x} \in i} P(c \mid \mathbf{x},i)}$$

(Weighted average of **all** regions vectors, **x**, in **all** images)

$$\mathbf{v}(c) = \frac{\sum_{i,\mathbf{x} \in i} (\mathbf{x} - \mathbf{u}).*(\mathbf{x} - \mathbf{u}) P(c \mid \mathbf{x},i)}{\sum_{i,\mathbf{x} \in i} P(c \mid \mathbf{x},i)}$$

(Weighted average of the squared deviations in **all** images).

where the .* notation is from Matlab, meaning element-wise.

Note that $P(w,i)$ and $P(\mathbf{x},i)$ are dropped from above three sums, as these are simply one for the words / regions that occur, and zero otherwise. Put differently, we only consider P(c|w,i) and P(c|x,i) for the regions and words that are in the image.

E-Step:

(The slight asymmetry here is because the model specifically translates from regions to words.)

$$P(c \mid \mathbf{x}) \quad P(\mathbf{x} \mid c) P(c)$$

(do for each image i)

To get the LHS, we have to normalize the RHS, so that the LHS sums to one over c for each x and for each i. However, doing this blindly likely will lead to computational issues. Thus see below for a way to get around this.

(This equation is referred to as +++ below)

$$P(c \mid X) \quad \sum_{\mathbf{x} \in i} P(c \mid \mathbf{x})$$

(do for each image i)

(This is from the model. We assume that all the regions for a given image combine via this sum to provide a "prior" on the concepts. This is the "translation" part of the model. Logically, once you form this sum, you should re-normalize so that for each X and for each i, the sum over c is one. However, the normalizing constant does not change during the EM iterations, so skipping this normalization is likely OK.

$$P(c \mid w, X) \quad P(w \mid c) P(c \mid X)$$

(do for each image i)

To get the LHS, we have to normalize the RHS, so that the LHS sums to one over c for each (w,i). Unlike the case for P(c|x), this can be done in the straightforward way (without scaling) as the numbers are already in a reasonable range.

Check and terminate:

Compute LL defined above and print it. It should go up at every step. You can stop when the successive differences in the LL is small; however, this typically over trains. For this assignment I suggest watching the LL for 30 iterations to make sure the program is working (and to provide the deliverable), but I suggest using the results after fewer (say 10) iterations to do the labeling. Pushing the number of iterations too high can lead to over-training

Computational notes

You will save time if you begin with a small data set and small values of M. You can simply truncated the input data to make it smaller. There is no point in running on a large data set until LL increases monotonically for a small data set.

Numerical problems abound in EM. I suggest you add 0.001 to each variance after computing it in the M-Step, especially if you are dividing by zero.

Once you work with a larger data set you will likely find that the quantities computed in the E step become too small (and thus become zero). The way to handle this is to do most of the E step calculations in log space. However you cannot compute sums in log space. In particular, you cannot compute the quantity labeled (***) in the equations for the log likelihood, and which is normalized in the E-step (+++). To do these sums, you first compute:

log(p(x,c)) = log(p32
(x|c)p(c)) = log(x|c) + log(p(c))

But now you are stuck. You have to exponentiate to get the sum. The trick is to first subtract the max over the clusters from the vector. Let:

$$M(x) = \max_c \{\log(p(x,c)\}$$

and define:

$$\log(\tilde{p}(x,c)) = \log(p(x,c)) - M(x)$$

alternatively:

$$\tilde{p}(x,c) = \frac{p(x,c)}{e^M}$$

To compute the normalization implied in (+++), you can use $\tilde{p}(x,c)$ instead of $p(x,c)$ because the e^M cancels. However, to compute (***), the e^M needs to be accounted for. We substitute

$$p(x,c) = e^M \tilde{p}(x,c)$$

into the log likelihood equation to get:

$$LL = \sum_{images} \left[\sum_{w \in image} \log(p(w \mid X)) + \sum_{x \in image} \log(\tilde{p}(x)) + M(x) \right]$$

The storage required for P(c|w,i) and P(c|x,i) can be very large as they grow with the number of images. The data for this assignment is small enough that you can simply pretend that this is not a problem. As an aside: In the case that P(c|w,i) and P(c|x,i) are too large, then we can do the E and M steps on each image in turn with appropriate extra initializations and cleanup steps. Computer vision students should know this trick, but it should not be necessary for this assignment.

How well are you doing?

This task is very difficult. Perfect results are not possible. However, your program should be able to get "sky" right most of the time, and good performance on flowers, cats, and planes. If you have accomplished this, and are still disappointed, consider what the comparison point would be.

It is beyond the scope of the requirements to provide performance numbers. However, in addition to visual inspection of results, then some indication of how well you are doing could be computed from the words provided in the optional file 2_discrete_item_sparse_mp. You can also compute the LL for the held out data. One of these methods could be used if you were trying to determine a good value for the number of concepts. However, note that these are image word prediction measures, not region prediction measures. In other words, these are proxy evaluations. The algorithm is based on the assumption that increasing region word prediction performance is the way to increase image word prediction performance.

Deliverables

Send me a tar'd or zipped directory which includes:

A README file explaining anything interesting about your work, and any comments.
A text file with the log likelihood for each iteration up to 30 for entire data set.
The Matlab files that you wrote
The text file which I can send to the labeling web page
Your 5 favorite labeled images and your 5 most problematic images (all from the test set).