GRB Classification - June 30, 2010

James Long

1 Introduction

I ran the June 6 analysis again with three changes:

1. For computing costs, cross validation was run 100 times. An average cost was computed, as well as the .05 quantile and the .95 quantile of the 100 trials.
2. Confusion matrices were computed.
3. Feature importance was assessed by determining the number of times each feature appeared in the trees. This is explained in more detail below.

There were also a few errors in my earlier code. Most significantly, I was computing the cost incorrectly before, so the previous cost results should be ignored.

2 Costs

Recall that the cost is computed using the formula,

$$\frac{\text{COST} * (\#\text{ high as low}) + 1 * (\#\text{ low as high})}{(\text{COST}*\#\text{ high}) + (\#\text{ low})} \tag{1}$$

The cross validated costs for each model are displayed in table 1. Recall that 6 models were constructed, without error features and with error features for misclassification cost of high as low of 3, 5, and 7. The first number in each cell is the mean cross validation error across the 100 cross validation samples. The numbers in parentheses are the 5^{th} smallest and 5^{th} largest CV estimate of error.

Some Observations:

1. The models that use error features perform better, in the case of Cost 5 and Cost 7, much better, than the models that do not use error features.
2. The loss goes up as the cost for classifying high as low increases. These values probably do not give a very good idea of what methods are working best since our measure for loss is a bit arbitrary. The confusion matrices in the next section give a better idea of how the methods are performing.

	Without Error Features	With Error Features
Cost 3	0.43 (0.34,0.51)	0.39 (0.29,0.51)
Cost 5	0.68 (0.58,0.77)	0.58 (0.44,0.72)
Cost 7	0.91 (0.78,1.04)	0.77 (0.62,0.92)

Table 1: Performance of CART for Two Sets of Features and 3 Loss Functions

3 Confusion Matrices

Confusion matrices for the three models without error features and the three models with error features are contained in tables 2 and 3 respectively. These were computed from the cross validation samples. So for each of the cross validation runs, a confusion matrix was constructed by classifying each observation using the model built when that observation was held out. The first number in each cell of the table is the average of the confusion matrices across the 100 cross validation runs. The numbers in parentheses are the .05 and .95 quantiles. These numbers should be conservative since the cross validated models use less data for construction and do not produce as good models as ones trained on all the data.

Some thoughts:

1. These numbers are not very encouraging. We never classify more than half of the high bursts correctly (lower right corner of each matrix is always less than 7).

2. I would like to know what Adam thinks of these numbers. For example in the model that uses error features and a cost of 7 (right side of table 3) if we followed the model we would likely follow up with telescope time on about 30 objects, 23 of which were actually low redshift and 7 of which were high redshift. We would miss following up on about 7 high redshift events. Is this reasonable performance? Increasing the cost to 10, we might expect to follow up on more like 40 objects, 32 of which were low redshift and 8 of which were high redshift. We would miss out on 6 high redshift events. Would this performance be better than with cost of 7.

3. The numbers follow patterns we expect. The number of misclassifications of low bursts goes up and the misclassification of high bursts goes down as the cost for misclassifying high as low increases (for both models with and without error features). This may mean that we can further reduce misclassification of high as low by increasing the cost to 10, 12, ect. As Tamara mentioned this approach to using cost as some sort of tuning parameter is a bit unsettling (also may get into overfitting issues here).

3.1 Without Error Features

	Cost 3		Cost 5		Cost 7	
	Low	High	Low	High	Low	High
Low	97.96 (94,103)	8.55 (6,11)	94.08 (91,98)	7.86 (5,10)	91.06 (87,95)	7.28 (5,9.05)
High	20.04 (15,24)	5.45 (3,8)	23.92 (20,27)	6.14 (4,9)	26.94 (23,31)	6.72 (4.95,9)

Table 2: Confusion Matrices from Cross Validation for Non Error Feature Model Using Costs of 3, 5, and 7. Columns are True Class, Rows are Estimated Class. The three sets of High/Low columns are for Costs of 3,5, and 7 (for High as Low). The numbers in parenthesis are the .05 quantile and .95 quantile for cross validation.

3.2 With Error Features

	Cost 3		Cost 5		Cost 7	
	Low	High	Low	High	Low	High
Low	100.44 (94,106)	9.18 (7,11.05)	97.93 (92.95,103.05)	8.13 (6,10.05)	95.21 (90.95,100)	7.28 (5,9.05)
High	17.56 (12,24)	4.82 (2.95,7)	20.07 (14.95,25.05)	5.87 (3.95,8)	22.79 (18,27.05)	6.72 (4.95,9)

Table 3: Confusion Matrices from Cross Validation for Error Feature Models Using Costs of 3, 5, and 7. Columns are True Class, Rows are Estimated Class. The three sets of High/Low columns are for Costs of 3,5, and 7 (for High as Low). The numbers in parenthesis are the .05 quantile and .95 quantile for cross validation.

4 Variable Importance Measures

I computed a measure of feature importance for each model. This was done as follows. For each of the 100 cross validation runs, 10 trees were grown each using 9/10 of the data (i.e. 10 fold cross validation). So for each model 1000 tree were grown in total. I counted the number of trees in which each feature appeared. This created an ordered list of which features appear in trees. I normalized these scores. Table 4 contains the feature importance information for the model built without error features using a cost of 7. Table 5 contains the feature importance information for the model built with error features using a cost of 7. Feature importance did not change significantly when cost was changed.

Some notes:

1. High importance means that CART used the feature a lot and that it is useful for classification. Low importance is a bit more difficult to interpret. It could be that the feature does not contain any useful information for classification. It could also be that the information in the feature is contained in another feature that ranks high for importance (as is the case with vmagisupper and whmagisupper). Or it could mean that the CART models are not making use of all information in the features.

2. For the without error features models there are three features that are used very frequently v_mag_isupper, A, and FLX_PC_LATE. Likewise for the models that use error features, except that GAM_PC_poserr is also important.

3. The without error features model make a bit of use of bat_rate_signif which should probably be considered an error feature (Adam?).

4. I used all error features when constructing the with error feature models. For some error features the poserr and negerr are identical so I should have used only one of the two.

5. It might be interesting to see how the with error features models are using GAM_PC_poserr. Any theories?

4.1 Without Error Features

Feature	Frequency
v_mag_isupper	0.26
A	0.24
FLX_PC_LATE	0.23
FLX_PC	0.04
xrt_column	0.04
NH_PC	0.03
wh_mag_isupper	0.03
EP0	0.02
NH_WT	0.02
bat_rate_signif	0.02
DT_MAX_SNR	0.01
FL	0.01
FLX_WT	0.01
bat_inten	0.01

Table 4: Variable Importance for Non Error Feature Model.

4.2 With Error Features

Feature	Frequency
v_mag_isupper	0.23
A	0.20
FLX_PC_LATE	0.20
GAM_PC_poserr	0.20
GAM_PC_negerr	0.03
wh_mag_isupper	0.03
FLX_PC	0.02
peakflux	0.02
EP	0.01
FLX_WT	0.01
NH_WT	0.01
xrt_column	0.01

Table 5: Variable Importance for Error Feature Model.

5 Some Thoughts on GRB Follow Up

In our last meeting, we discussed a bit about classification, prediction, loss functions, and precisely what our goal is. Over the past few weeks I've done some thinking (still far from complete) on these issues. Here are my thoughts.

Adam needs to decide whether to follow up a GRB detection with telescope resources as the bursts arrive. Our statistical methods may produce a classification, a prediction or a probability density but what we ultimately need is a decision to follow up or not follow up.

While in practice we need to make a decision for follow up immediately after detection, consider the simplified senario where we detect n GRBs with Swift and we have the resources to follow up on n_0 where $n_0 < n$. Presumably there is some loss function that expresses our level of dissatisfaction in the choice we make (follow up / do not follow up) based on the actual redshift of the GRB. In mathematics a function with domain decision cross redshift and range the real numbers. So,

$$l : D \text{ x } R \rightarrow \mathbb{R} \tag{2}$$

where D is our decision, 1 for follow up, 0 for not follow up. R is the actual redshift of the GRB (unknown). $\mathbb{R}$ is the set of real numbers. So a few values of l could be,

$$l(1, 2.5) = 2$$
$$l(1, 8) = 0$$
$$l(0, 3) = .5$$
$$l(0, 9) = 30$$

These four plausible losses express the ideas that: if we follow up and the redshift is low (2.5) we lose a little (2), if we follow up and the redshift is high (8) we lose nothing, if we do not follow up and the redshift is low (3) we lose very little (.5), and if we do not follow up and the redshift is high (9) we lose a lot (30). Adam probably has an intuitive idea for what this function should look like. It may change for different applications, but it does not depend on the data.

For each of the n GRBs we observe we have features. Call the set of features associated with GRB i, X_i. So X_i contains whmagisupper, T90, peakflux, ect. measurements for the i^{th} burst. In an ideal world we would know the probability density $p(r|X)$, the probability that a GRB with features X has redshift r. In practice we can estimate $p(r|X)$ from the training data. Call this estimate $\hat{p}_X(r)$. There are many statistical methods that can be used to create $\hat{p}_X$. For example,

1. With ordinary linear regression $\hat{p}_X$ is normal density with the same variance for all X. The mean of the density changes in a linear manner with the predictor variables (features).

2. With regression trees for CART $\hat{p}_X$ is piecewise contant across the domain X (technically CART only says the mean of $\hat{p}_X$ is piecewise constant, but we would probably make the entire function constant). We could probably then assume that the density is normal (with mean given by CART) and estimate the variance as the variance of the observations in the given level. We could also run a kernel density estimate on the redshift values in each set of the partition.

3. I believe that the Gaussian Processes that Tamara discussed output a $\hat{p}_X$.

Since we do not have a ton of data $\hat{p}_X$ will most likely not get too exotic. Also note that since $\hat{p}_X(r)$ is a density there are clear restrictions on its form, such as is takes positive values for all X and r as well as integrating across r to 1, i.e.

$$\int_0^\infty \hat{p}_X(r)dr = 1$$

for all X. For each of the n GRBs we can compute the expected loss for not following-up and for following up. For GRB i this is,

$$L_{X_i}(1) = \int_0^\infty l(1,r)\hat{p}_{X_i}(r)dr \tag{3}$$

$$L_{X_i}(0) = \int_0^\infty l(0,r)\hat{p}_{X_i}(r)dr \tag{4}$$

Then we compute the difference in loss for not following up versus following up the i^{th} GRB. So,

$$\Delta L_{X_i} = L_{X_i}(0) - L_{X_i}(1)$$

Then we can order the ΔL_{X_i} from greatest to smallest. We follow up on the largest n_0 of the ΔL_{X_i} values.

5.1 Ongoing Thinking

Some additional thoughts:

1. The situation we are facing must be a fairly common occurance and I am sure there is already some well established literature on the subject. We should look for this.

2. The problem I addressed is a bit simplier than actual GRB followup since in practice follow up is done in real time. There are a few ways to deal with this. Here is one: We have built a model on training data and ordered the ΔL_{X_i} for the training set. Say we expect to get 20 GRBs detections in a 30 day period and we have viewing time to follow up on 15 detections (this would all be determined by the history of detecting GRBs and the restrictions on viewing time). If on day 1 we get a GRB detection we calculate ΔL_{X_i} for the GRB and if it is higher than the first quartile of the ordered list from the training set we follow up. This is because we expect 20 GRBs for the month, so we want to follow up on three fourths of the ones we see, so if we see something that is more desirable to follow up on than the bottom fourth of the training set we go ahead and follow up. Say we get through half the 30 day period and we have followed up on 10 GRBs already. We only have time to follow up on 5 more. We expect to see 10 more GRBs in the next 15 days. So now if we get a detection we follow up only if ΔL_{X_i} for the new burst is in the top half of the ordered list of training data. The basic idea is that we compare ΔL_{X_i} for new detections to the training data, taking account for how much viewing time we have and how many detections we expect to get.

3. One issue that is still bothering me is the role of a loss function in constructing $p_X(r)$. In some ways I would like the loss function to only enter the problem when computing the L_{X_i} in equation (3). However CART and most other techniques for constructing $p_X(r)$ use a loss function somewhere. I am wondering about whether the loss function chosen for computing (3) somehow induces an ideal loss function for CART. This is still very fuzzy in my mind.