Classifying the supernova data

Charlotte Wickham

May 10, 2007

Supernova data

- Collection of 5000 supernova and 5000 other objects
- 19 features transformed by Raquel
- Split into balanced training and test sets. 9000 in training set, 1000 in test set.

Summary

Error rates:

1. Support vector machines $\approx 5\%$
2. Neural Networks $\approx 5\%$
3. Random Forests $\approx 5\%$
4. Bagged Trees $\approx 6\%$
5. Classification Trees $\approx 8\%$
6. Boosted Trees $\approx 9\%$

Outline

Training svm

- Kernel Function
  Radial basis function kernel

$$\exp(-\gamma|u - v|^2)$$

- 10 fold cross validation to search for good parameters
- Parameters in kernel function
  γ: 14 values between 0.0001 and 5
- Tuning parameter (how much misclassification are we allowing?)
  search over 10 values between 0.5 and 100

Best Support Vector Machine

$\gamma = 0.05$ cost $= 9$

		Prediction	
		Other	Supernova
Actual	Other	484	16
	Supernova	32	468

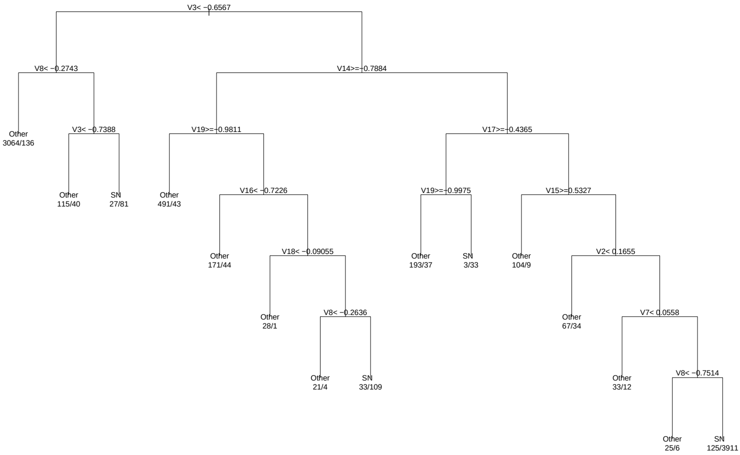
Error $= 4.8\%$

Outline

Classification Trees

- Grow full size tree - will tend to over fit
- Prune back using cross validation

Pruned tree - has 14 splits



Left branch

Splits on two variables classified a lot of the "Other" objects. They were:

- perinc (V3) - % flux increase in aperture from REF to NEW
- neighbordist (V8) - distance to the nearest object in REF

Performance on test set

- Classification Tree

		Prediction	
		Other	Supernova
Actual	Other	461	39
	Supernova	45	455

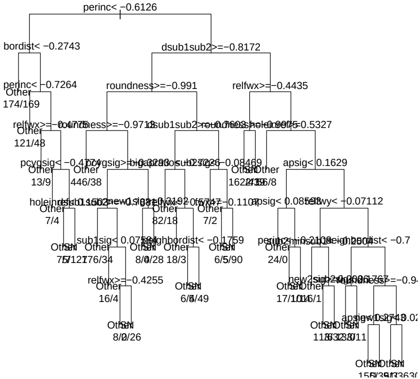
Error $= 8.4\%$

Adding costs and priors

- See a lot more other objects than supernova
- Would like to be more accurate identifying non supernova to minimize false discoveries.
- Trees have the option of accounting for this by adding priors or misclassification costs.
- Priors didn't work too well

Costs in supernova data

Costs: $C(SN|Other) = 2, C(Other|SN) = 1$



		Prediction	
		Other	SN
Actual	Other	476	24
	SN	55	445

Total error $= 7.9\%$

$+$ve's $= 11\%$

-ve's $= 4.8\%$

Outline

Bagging for trees

1. A bootstrap sample, $\mathcal{L}_B$, from $\mathcal{L}$ is selected.
2. A tree is grown on $\mathcal{L}_B$ (and $\mathcal{L}$ is used to choose a pruned subtree).
3. This is repeated K times to give a sequence of predictors, $\phi_1(x), \ldots, \phi_K(x)$.
4. The bagged predictor of, y_n, is $\text{avg}_k \phi_k(x_n)$ for regression trees or is the class having the plurality in $\phi_1(x), \ldots, \phi_K(x)$ for classification trees.

Note: Bagging isn't restricted to trees.

Bagging Supernova data

		Prediction	
		Other	SN
Actual	Other	477	37
	SN	23	463

Total error = 6%
+ve's = 11.6%
-ve's = 4.6%

Outline

Random input selection

Fix parameter K

- Draw a bootstrap sample from the training set
- Grow full size tree as usual except at each node choose K variables randomly on which to search for the best split.
- Repeat N times to generate N trees.

Breiman suggests $K = \sqrt{\text{number of variables}}$.

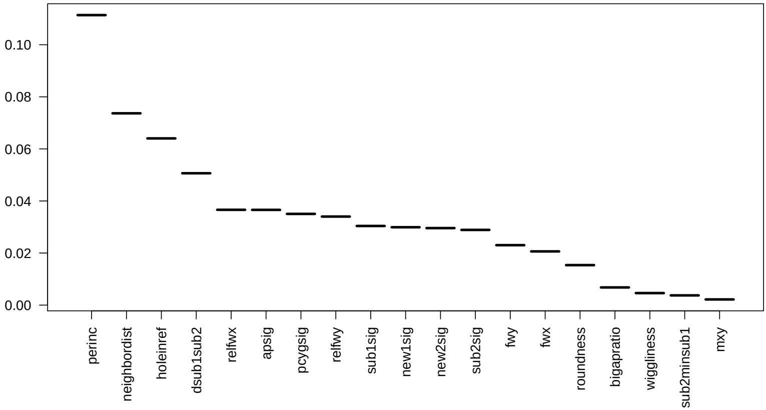
Performance on test set

K = 4

		Prediction: Other	Prediction: Supernova
Actual	Other	484	16
	Supernova	39	461

Error: 5.5%
(were getting about 8% from CART and 6% from bagging.)

Supernova - Variable Importance



Supernova Data - Try different K

- $K = 2$

		Prediction	
		Other	Supernova
Actual	Other	484	16
	Supernova	38	462

Error: 5.4%

- $K = 8$

		Prediction	
		Other	Supernova
Actual	Other	480	20
	Supernova	37	463

Error: 5.7%

Outline

Boosting

Idea in the binary classification context ($Y \in \{-1, 1\}$).

- Take a weak classifier (one that does just a bit better than random guessing).
- Fit the classifier to the data to get $G_1(X)$.
- Give more weight to the observations in the training set it gets wrong.
- Re-fit the classifier to the reweighted data.
- Repeat M times to get a sequence of classifiers, $G_m(X)$.
- The predicted value of a new data point is a **weighted** sum of classifiers.

$$G(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

AdaBoost Algorithm

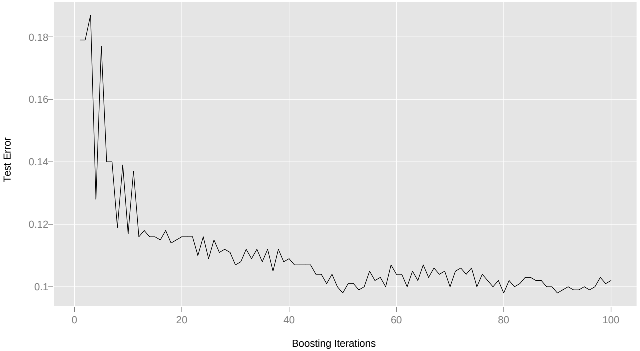
- Gives more weight to classifiers in the sequence with low training error.
- Gives more weight to observations that are misclassified. The better the classifier overall the bigger the weight on the misclassified observations.
- Choice of classifier is free. Often use trees with very few splits. Stump = tree with one split.

Performance on test set

- Trees with one split (stumps)

		Prediction: Other	Prediction: Supernova
Actual	Other	451	49
Actual	Supernova	47	453

Error: 9.6%



Got about 8% from CART and 6% from bagging and 5% from random forests.

- Trees with two splits

		Prediction	
		Other	Supernova
Actual	Other	455	45
	Supernova	45	455

Error: 9.0%

- Trees with three splits

		Prediction	
		Other	Supernova
Actual	Other	455	45
	Supernova	47	453

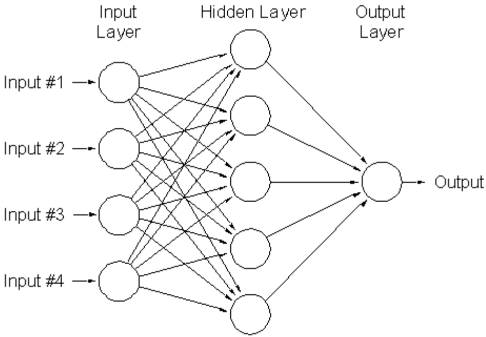
Error: 9.2%

Outline

Neural Networks

Simplest case

- Have p inputs $\boldsymbol{x}$
- Have one hidden layer with each unit being a function (ϕ_j) of a linear combination of the inputs.
- Each output is a function (ϕ_0) of a linear combination of the hidden units.



In practice

- Scale all variables to have mean 0 variance 1 - to ensure input treated equally in regularization.
- Choose decay parameter by cross validation.
- Number of hidden units generally doesn't matter as long as it is big enough.
- Can be sensitive to initial conditions. Can average a few instances.

Cross validation

```
      size decay      fit
 [1,]    5 0.010 4.333333
 [2,]    5 0.050 4.377778
 [3,]    5 0.001 4.600000
 [4,]   10 0.010 4.133333
 [5,]   10 0.050 3.977778
 [6,]   10 0.001 4.411111
 [7,]   15 0.010 4.366667
 [8,]   15 0.050 4.322222
 [9,]   15 0.001 4.388889
[10,]   20 0.010 4.422222
[11,]   20 0.050 4.433333
[12,]   20 0.001 4.511111
```

Fit neural net with 10 hidden units and decay 0.05. Repeat five times and average result.

Supernova data

- Disagreement between repetitions

		NN 1	
		Other	Supernova
NN 2	Other	495	15
	Supernova	10	480

2.5% disagreement

- Prediction error

		Prediction	
		Other	Supernova
Actual	Other	477	23
	Supernova	28	472

5.1% error

Summary

Error rates:

1. Support vector machines $\approx 5\%$
2. Neural Networks $\approx 5\%$
3. Random Forests $\approx 5\%$
4. Bagged Trees $\approx 6\%$
5. Classification Trees $\approx 8\%$
6. Boosted Trees $\approx 9\%$

Variable name	Description
apsig	signal-to-noise ratio in aperture
perinc	% flux increase in aperture from REF to NEW
pcygsig	difference of flux in 2*FWHM of aperture and 0.7*FWHM; detects misaligned REF and NEW images)
mxy	x-y moment of candidate
fwx	FWHM of candidate in x
fwy	FWHM of candidate in y
neighbordist	distance to the nearest object in REF
new1sig	signal-to-noise of candidate in NEW1
new2sig	signal-to-noise of candidate in NEW2
sub1sig	signal-to-noise of candidate in SUB1
sub2sig	signal-to-noise of candidate in SUB2
sub2minsub1	weighted signal-to-noise difference between SUB1 and SUB2
dsub1sub2	difference in pixel coordinates between SUB1 and SUB2 (motion measurement)
holeinref	measure of negative pixels on REF in region of candidate
bigapratio	ratio of sum of positive pixels to sum of negative pixels within aperture
relfwx	REF image FWHM in x divided by NEW image FWHM in x
relfwy	REF image FWHM in y divided by NEW image FWHM in y
roundness	object contour eccentricity; ratio of powers in lowest order negative and positive Fourier contour descriptors
wiggliness	object contour irregularity; power in higher order Fourier contour descriptors divided by total power

Random Forests variable importance

- Use out of bag observations and randomly permute one variable.
- Run the observations down the tree and record classification.
- Repeat for each tree.
- Compare the misclassification rate with the noised up variable to the out of bag estimate without permutation.
- Variable Importance = percent increase in misclassification.