Classification Overview

Jared S. Murray

Introduction

- Moving from *regression* (predicting a number) to *classification* (picking a category) is relatively straightforward
- We still evaluate methods based on out-of-sample performance measures using cross-validation/data splitting
- For tree models, we grow our trees in (almost) the same way
    - Single tree/random forests: Use different error measures
    - Boosting: Similar in spirit for classification (construct a series of weak learners sequentially and combine) but the details are complicated (see ch 10 of Elements of Statistical Learning)

Tree splitting criteria: Gini Index

- When growing a tree, we prefer predictive splits. Predictive splits $==$ low variance of Y in the leaves $==$ low MSE!
- For categorical Y, we want leaves to have *homogenous* Y values

Tree splitting criteria: Gini Index

- Recall that for a binary random variable X with $\Pr[X = 1] = p$,

$$\mathrm{Var}[X] = p(1 - p)$$

- Gini index: Total leaf-wise variance of k dummy variables $D_1, \ldots D_k$ (where $D_k = 1$ if $Y = l$ and zero otherwise)
  - Want to minimize!
- In a leaf indexed by m for a Y with k levels,

$$G = \sum_{k=1}^{K} \hat{p}_{mk} \left(1 - \hat{p}_{mk}\right)$$

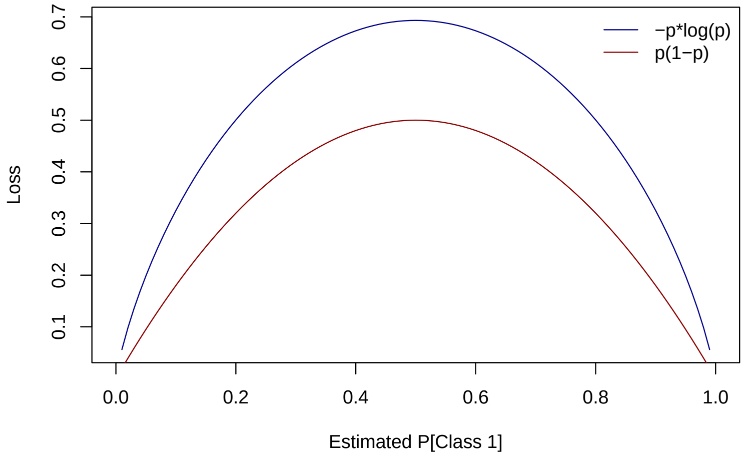
Tree splitting criteria: Cross-entropy

- Cross-entropy is another measure of "homogeneity" we can minimize
- Cross entropy is given by:

$$E = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

- Equivalent to the in-sample mean log loss (more soon)
- Motivated by maximum likelihood estimation (more soon about this too)
- For now just note that they're very similar measures...

Tree splitting criteria: Comparison for binary classification



Measuring performance in classification

- Whether we use trees or another method, we need new measures for (out-of-sample) performance.
- We'll stick to *binary* classifiers here; for multiclass things get more complicated
    - Or not: Pick one category and compare "one vs all"

The confusion matrix

An example from your textbook:

		True default status		
		No	Yes	Total
Predicted default status	No	9644	252	9896
	Yes	23	81	104
	Total	9667	333	10000

Generically:

		True category		
		0	1	Total
Predicted category	0	True Negatives	False Negatives	–
	1	False Positives	True Positives	–
	Total	–	–	–

Error rates

- This single 2x2 table can be turned into zillions of error measures!
- See this Wikipedia summary for example
- All have pros and cons; we will focus on the most common measures

Measuring classification quality

		True default status		
		No	Yes	Total
Predicted	No	9644	252	9896
default status	Yes	23	81	104
	Total	9667	333	10000

- Accuracy: $\frac{9644+81}{10000}$
- True Positive Rate: $\frac{81}{81+252}$ (aka recall, sensitivity)
- False Positive Rate: $\frac{23}{23+9644}$ (aka 1-specificity)
- Precision: $\frac{81}{81+23}$ (aka positive predictive value, 1 - false discovery rate)
- Negative predictive value: $\frac{9644}{9644+252}$

Measuring classification quality

		True default status		
		No	Yes	Total
Predicted default status	No	9644	252	9896
	Yes	23	81	104
	Total	9667	333	10000

- True Positive Rate: How accurately do I classify truly positive cases?
- False Positive Rate: How **in**accurately do I classify truly **negative** cases?

With class imbalance, overall accuracy can be high while TPR is very low (or FPR is very high)

Measuring classification quality

		True default status		
		No	Yes	Total
Predicted	No	9644	252	9896
default status	Yes	23	81	104
	Total	9667	333	10000

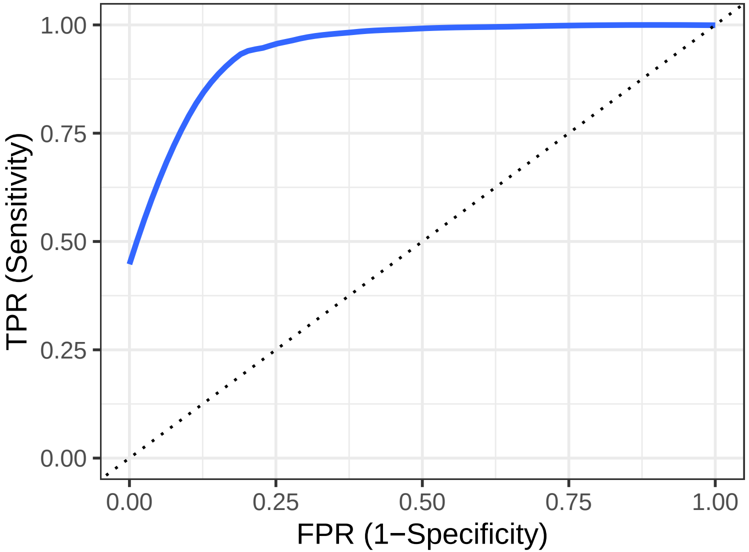
- Precision: **Of the cases classified as positive**, how many are truly positive?
  - Think about a disease test: This is Pr[sick | positive test]
  - TPR is Pr[positive test | sick]
- Which statistic(s) to optimize for depends on context.

From classifiers to classifications

- In the preceding example we took as given the classifications from a model/algorithm
- In practice, the mapping from classifier -> classifications isn't automatic:
  - Single tree models and boosted tree models return estimated class probabilities
  - So can random forests, or they return vote shares (e.g. 65% of the trees predicted class 1 and 35% predicted class 0)
- Most classifiers will produce a predicted probability, or at least a score like RF vote shares – call this $\hat{p}$
- Classification rules are given by "Predict 1 if $\hat{p} >$ s"
  - So a single model fit will give a family of classifications that trade off false positives and false negatives, by making more ($s \to 0$) or fewer ($s \to 1$) positive (class $= 1$) predictions.

ROC curves

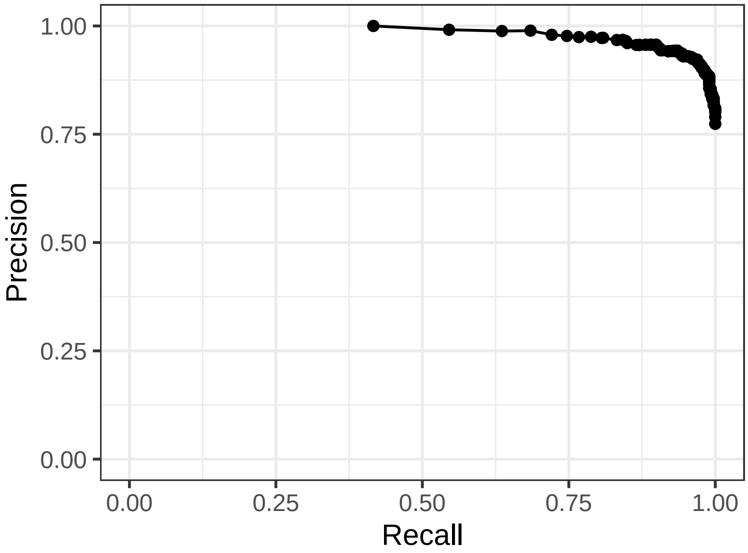
Varying s from 0 to 1 gives us the receiver operating characteristic (ROC) curve:



ROC curves

- The 1-1 line represents the ROC curve when predicted probabilities are completely random noise
- The *area under the ROC curve* (AUC) is a measure of classifier performance
  - AUC = 1 if and only if there is a threshold that gives perfect classification rule
  - AUC is the probability that a randomly selected positive case has a higher score (predicted probability) than a randomly selected negative case
- Youden's J is TPR + (1-FPR) - 1
  - For any threshold value this is the distance from the point on the ROC curve to the 1-1 line
  - Aka “informedness”, a measure of improvement over random guessing
  - Without additional considerations, not the worst way to pick a threshold

Precision - Recall curves



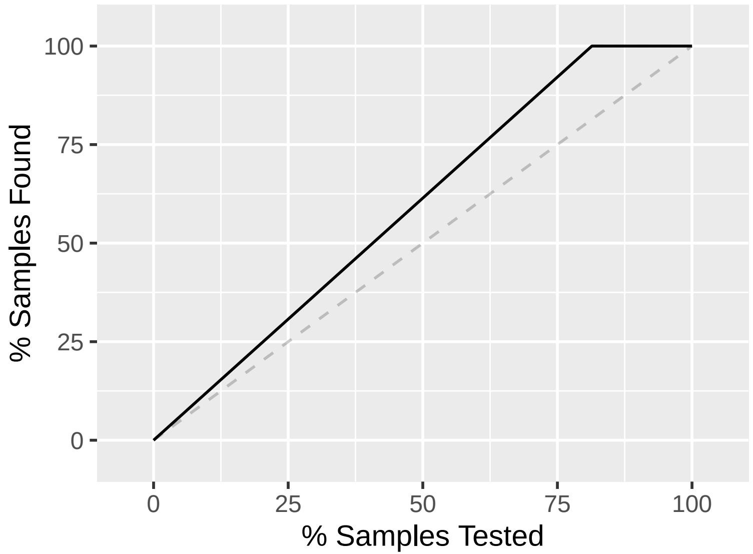
Precision - Recall curves

- Precision - recall curves summarize performance on the *positive* class
- A point on the PR curve tells us that if we set a threshold to correctly classify 100R% of the true positive cases then at most 100P% of the cases we predict positive will in fact be positive
- Precision $= 1$ and Recall $= 1$ means we classify all true positives as positives **without** classifying any true negatives as positives
- The PR curve is (approximately) flat at the prevalence (proportion of true positive cases) when predicted probabilities are completely random noise
- Popular in information retrieval contexts, where a "positive" case is a relevant result and we want to return as many as possible without returning too many irrelevant results

Lift curves

- Lift (gain) curves plot TPR against the percentage of cases classified as positive
- Random guessing == lift curve that is the 1-1 line
- A more intuitive description of the lift curve:
  - Imagine I use my classifier to decide who will see an ad by predicted clickthrough rates
  - I use my predicted probabilities of clickthrough to target some fraction of total users (based on budget)
  - The lift curve says: If I target this way, showing X% of users the ad will find Y% of the users who would click through (true positives)

Lift curves



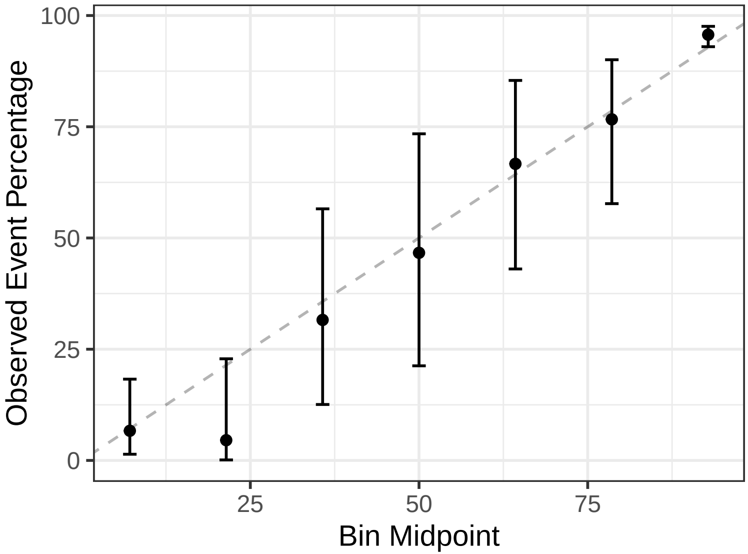
Calibration

- A model is *calibrated* if its predicted *probabilities* are accurate
    - If we collected all the cases where the pred prob is p, 100p% of them would actually be positives
- A calibrated model is not necessarily a good model!
    - If 3% of the cases are truly positive, a model that predicts $\hat{p}_i = 0.03$ for every case is calibrated
    - We also want **sharpness**: $\hat{p}_i \approx \Pr[Y = 1 \mid X = x_i]$, i.e., predictions should vary with covariates accurately
- Calibration is not necessary for accurate classification, but it can be important in decisionmaking
    - For example, using predicted default probability to calculate expected profit/loss from writing the loan

Calibration plots

- Calibration plots check for calibration this approximately:
  - Bin the predicted probabilities (x axis)
  - Compare the bin midpoint to observed proportion of positives (y axis)
  - Account for uncertainty in observed proportions with a confidence interval
- y=x line should be close to the points and inside the interval across bins

Calibration plots



Out-of-sample loss functions for training

- What (out-of-sample) error/accuracy metrics should we use to compare competing models?
- It depends! Common choices:
  - Accuracy (easy to understand)
  - Cohen's kappa (increase in accuracy over guessing, better than accuracy for imbalanced datasets)
  - Area under ROC curve (accounts for varying classification rules)
  - Log loss: $-\sum_{i=1}^{n} \log[\hat{p}_{y_i}]$, where $\hat{p}_{y_i}$ is the predicted probability of the *observed* class for observation i (encourages calibrated predicted probabilities and accurate classifications).
- These will often (but not always!) give the same or very similar model rankings