Classification: Random Forests Wrap Up + Understanding Classifiers

Siva Balakrishnan
Data Mining: 36-462/36-662

February 26th, 2019

Mainly K&J Chapter 11

Quick Announcements

- Riccardo's office hours.

Recap: Ensemble Classifiers

- Construct many different classifiers $\{\widehat{f}_1, \ldots, \widehat{f}_b\}$. Combine their predictions by taking a (weighted) majority vote:

$$\widehat{f}(x) = \text{sign}(\sum_{t=1}^{b} \alpha_t \widehat{f}_t(x)),$$

  for some weights α_t.

Recap: Gradient Boosting for Regression

1. Key idea: Suppose we want to do regression. Fit an ensemble of regression trees, incrementally, i.e.:

$$\widehat{f}(x) = \sum_{b=1}^{B} \alpha_b \widehat{f}^{(b)}(x)$$

   where each $\widehat{f}^{(b)}(x)$ is a regression tree.

2. We simply fit a small regression tree (a stump), find the residual

$$R(x_i) = y_i - \widehat{f}^{(1)}(x_i)$$

   and now fit a new tree to the residuals, i.e. replace y_i with $R(x_i)$, and repeat.

3. In practice, this is a very popular and powerful regression algorithm.

Recap: Bagging

Given a training data (x_i, y_i), $i = 1, \ldots n$, bagging[1] averages the predictions from predictors (here trees) over a collection of boostrap samples.

For $b = 1, \ldots B$ (e.g., $B = 100$), we draw n boostrap samples (x_i^{*b}, y_i^{*b}), $i = 1, \ldots n$, and we fit a classification tree $\hat{f}^{\text{tree},b}$ on each sampled data set.

To classify an input $x \in \mathbb{R}^p$, we simply take the most commonly predicted class:

$$\hat{f}^{\text{bag}}(x) = \underset{k=1,\ldots K}{\text{argmax}} \sum_{b=1}^{B} 1\{\hat{f}^{\text{tree},b}(x) = k\}$$

This is just choosing the class with the most votes.

[1] Breiman (1996), "Bagging Predictors"

Recap: Estimating Class Probabilities

- Each tree gives us a set of predicted class probabilities at x:

$$\widehat{p}_k^{\text{tree},b}(x),$$

  These are simply the proportion of points in the appropriate region that are in each class.
- Alternate bagging method: given an input $x \in \mathbb{R}^p$, instead of simply taking the prediction $\widehat{f}^{\text{tree},b}(x)$ from each tree, we go further and look at its predicted class probabilities $\widehat{p}_k^{\text{tree},b}(x)$, $k = 1, \ldots K$, and combine those.
- We define the bagging estimates of class probabilities:

$$\widehat{p}_k^{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \widehat{p}_k^{\text{tree},b}(x) \quad k = 1, \ldots K.$$

Recap: Random Forests – Improved Bagging

Random forests[2], an incredibly popular and successful prediction approach, are a simple modification of bagged trees.

We've seen that bagging should improve with independence of the trees. Bagged trees can have dependence because the same strong variable always makes it into the same split.

Random forests make the trees more independent by only allowing a small subset of variables to be considered ***at each split***. At each split, $m < p$ variables are selected, and a split is chosen from among those variables.

This leads the trees to be more varied and less independent. It also leads to a nice balance among correlated variables. Finally, it actually gives better predictions!

[2] Breiman (2001), "Random Forests"

How do we estimate error?

How can we estimate how well we will predict/classify on new data?

Cross-validation is tempting, because we have no creativity.

However, cross-validation could be quite expensive, since we're boostrapping and developing hundreds of trees each time.

What about the samples each bootstrap did not select?

Out-of-Bag error estimation

Each training point should be missing from about $1/3$ of the bootstrap samples. That subset of the trees have not been overfit to the training point!

For each sample point, we get an average prediction from the $1/3$ of the trees that didn't include it. We combine these predictions over all the points to get an out-of-bag MSE estimate.

This acts like a cross-validated estimate, but it's free!

Digging into Random Forests

The bagged classifier is now an average of many trees. This is much harder to interpret! (If you thought trees were hard to interpret...)

Similarly, one of the big selling points of the Lasso is that it produces *interpretable* models. By seeing the sparsity pattern, we know which variables are being used for predicting.

We see that random forests predict incredibly well. However, an average of hundreds of random trees is harder to understand.

We need tools to peek into these "black box" methods to understand how they're using the data.

Variable Importance

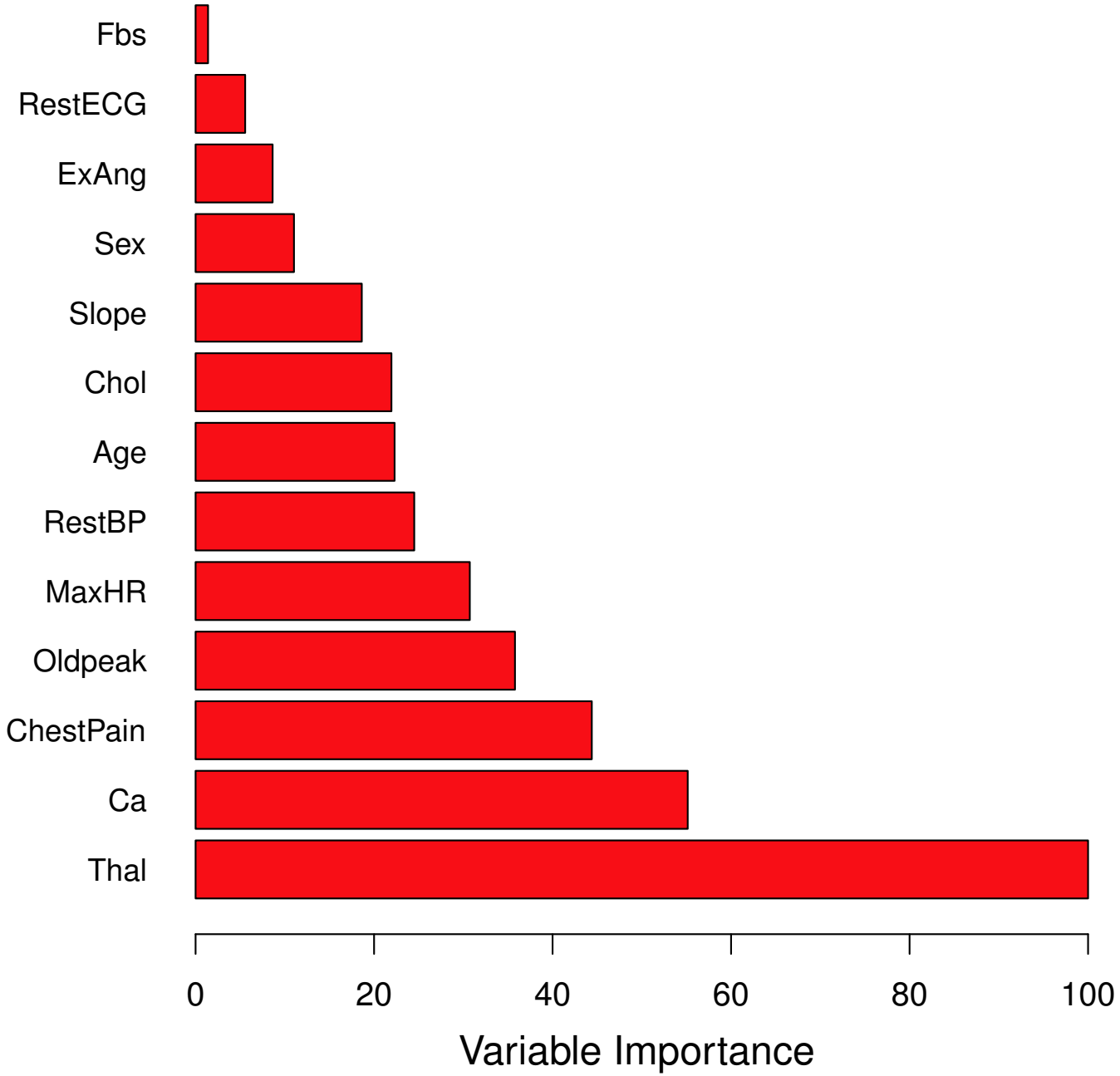
The most basic task is to understand which predictors are important for making the random forest prediction.

Random forests don't explicitly do variable selection like the lasso, but their individual trees tend to rely more on informative variables when they search for the next split.

There are two popular ways to measure variable importance:

1. For each variable, measure the amount that RSS (or Gini index) decreases due to splits in that variable. Average this over all trees in the forest
2. Randomly permute each variable (one at a time) and see how much the model performance decreases.

The `varImpPlot` method in `R` computes the first of these.


Fbs
RestECG
ExAng
Sex
Slope
Chol
Age
RestBP
MaxHR
Oldpeak
ChestPain
Ca
Thal
0
20
40
60
80
100
Variable Importance

Disadvantages

It is important to discuss some disadvantages of bagging:

- *Loss of interpretability:* the final bagged classifier is not a tree, and so we forfeit any interpretability a tree might have had.

- *Computational complexity:* we are essentially multiplying the work of growing a single tree by B (especially if we are using the more involved implementation that prunes and validates on the original training data)

Thinking About Classifiers

- So far, we have focused mainly on the 0/1 loss and using the misclassification error (on a held-out set) to evaluate our classifiers. In practice, there are many more quantities that we can compute to assess a classifier.
- Broadly, can separate into two kinds of measures:
  - Assessing the predicted class labels (things like classification error)
  - Assessing the predicted class probabilities (things like ROC curves, calibration etc.)

Calibration

Is our $\widehat{p}(x)$ a good estimate of $p(x) = \mathbb{P}(Y = 1|X = x)$?

Why do we care? We're just going to threshold it...

There is information in knowing $\mathbb{P}(Y = 1|X = x)$.

- A case that is definitely fraud ($p(x) = 0.99$) may be handled very differently than a case that might be fraud ($p(x) = 0.52$).

  Calculating the proper trade-offs for making decisions depends on the probabilities, not just the categories.

Note: Even if $\widehat{p}(x)$ is not a good approximation of $p(x)$, thresholding $\widehat{p}(x)$ might still give good guess at categories.

Calibration: Aside

- Many classifiers (SVMs, Boosting, Neural Networks) do not directly produce estimates of the class conditional probabilities. Rather, they just produce scores which we then threshold.
- In practice: we can transform scores $\{y_1, \ldots, y_K\}$ for a point x into numbers that behave like probabilities (they still are not really probabilities).
- Often:

$$\mathbb{P}(y = k|X = x) = \frac{\exp(y_k)}{\sum_{j=1}^{K} \exp(y_j)}.$$

Calibration plots

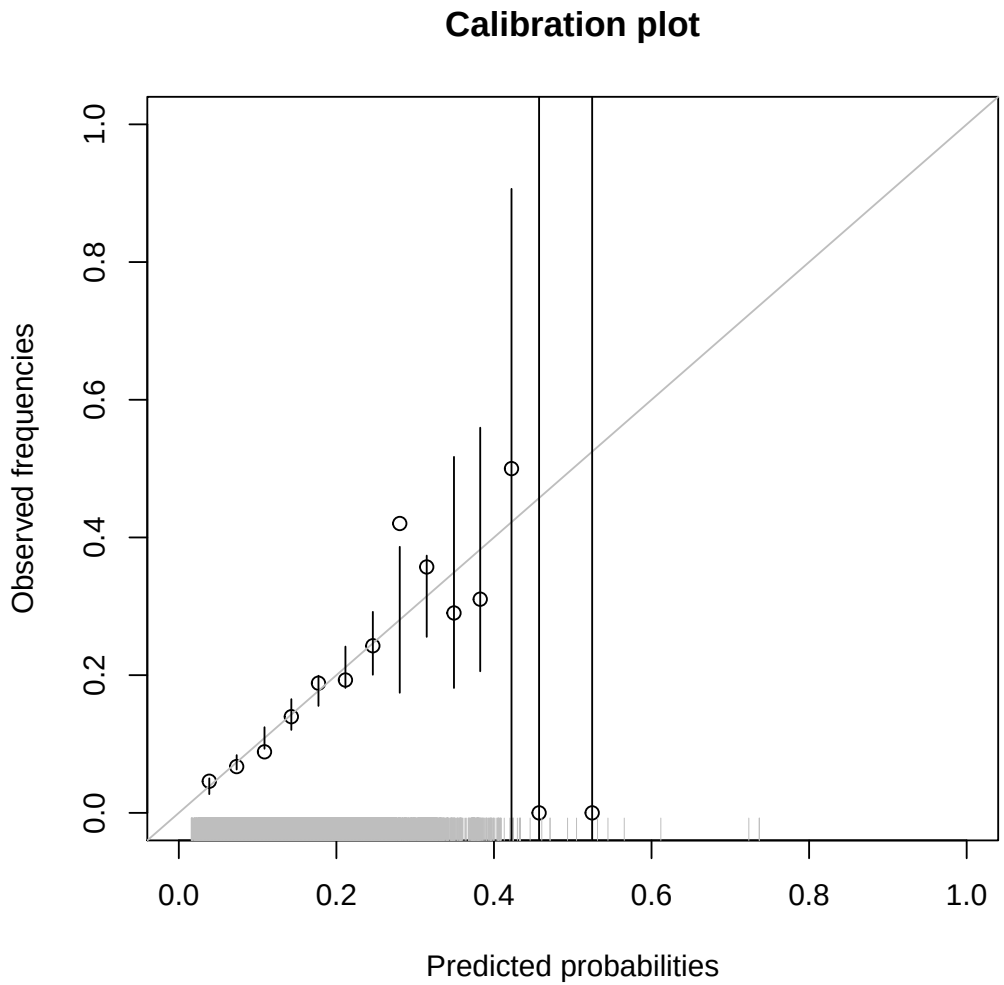
Focus for now on the binary case. Let us denote $\widehat{p}(x) := \widehat{\mathbb{P}}(y = 1|X = x)$. To check how well calibrated your $\widehat{p}(x)$ is, you can make a *calibration plot*:

1. Bin the data according to predicted probability, $\widehat{p}(x)$. E.g., $[0, 0.1], (0.1, 0.2], \ldots, (0.9, 1]$.

2. For each bin, calculate the proportion of observations with class $Y = 1$

3. Plot these true fractions against the midpoints of the bin predicted probabilities

Calibration plots



Calibration plots

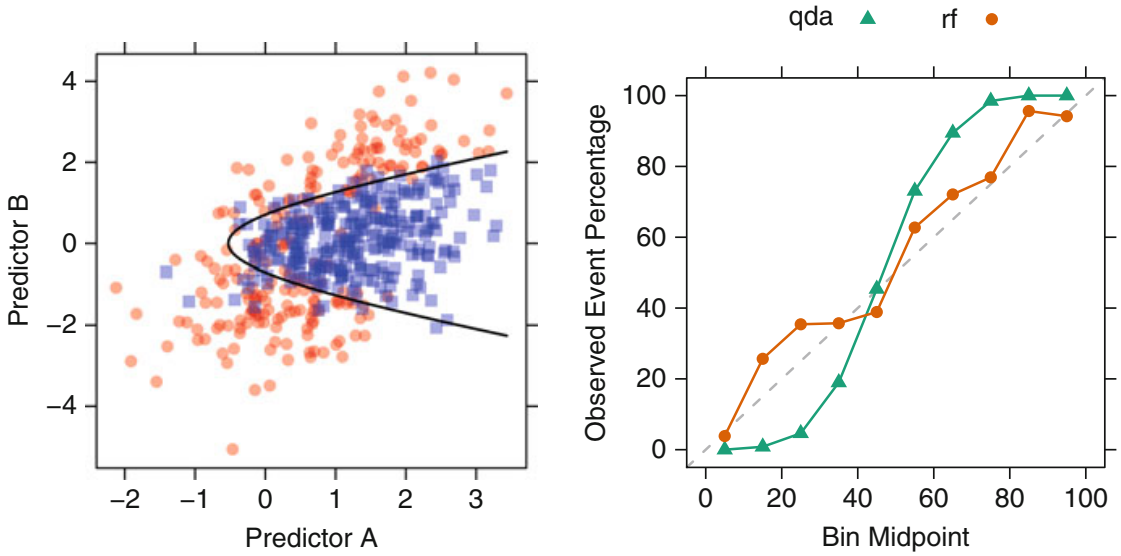


Fig. 11.1: *Left*: A simulated two-class data set with two predictors. The *solid black line* denotes the 50 % probability contour. *Right*: A calibration plot of the test set probabilities for random forest and quadratic discriminant analysis models

(From *Applied Predictive Modeling* by Kjell Johnson and Max Kuhn)

Miscalibration

- When the calibration is off you can try to fix it by transforming $\widehat{p}(x)$.
- The usual way to do this is to run a logistic regression with $\widehat{p}(x)$ as the only feature.

Confusion matrix

Back to classifying

- $Y = 1$ if an event of interest happened
- $Y = 0$ if it did not happen

Recall: In K-class classification, we can make $K(K-1)$ different kinds of mistakes!

We will be thinking more about these confusion matrices.

In our 2×2 setting:

	True $Y = 1$	True $Y = 0$
Guess $Y = 1$	True Positive	False Positive
Guess $Y = 0$	False Negative	True Negative

Misclassification rate

Suppose that I tell you my classifier has 97% accuracy. Is this good?

It depends! What if I am predicting lottery wins. The probability of winning may be $1/175000000$.

In that case, guessing "no" has a misclassification rate of 5×10^{-9}...

We need to consider the *base rate* or the error rate of the naïve classifier. This depends on the probability of each class given no further information.

Confusion matrix: error rates

	True $Y = 1$	True $Y = 0$
Guess $Y = 1$	True Positive	False Positive
Guess $Y = 0$	False Negative	True Negative

Other error rate measures better capture asymmetric concerns when classifying:

- Sensitivity (Recall, Hit rate):

- Specificity (True Negative Rate, TNR):

These quantities will allow us to better think about cost trade offs.

Sensitivity and Specificity

Sensitivity: Fraction of points with $Y = 1$ that we find.

- Want high Sensitivity when False Negatives are more costly than False Positives. Example: Fraud detection.

Specificity: Fraction of points with $Y = 0$ that we avoid.

- Want high Specificity when False Positives are more expensive than False Negatives. Example: Criminal trials.

We would like both of them to be high, but we need to make tradeoffs as we adjust our threshold.

- We can flag more cases as fraud to improve sensitivity, but this will decrease our specificity.

Other metrics

Positive predictive value (PPV):

$$PPV = \frac{\#\text{ observations correctly classified as positive}}{\#\text{observations classified as positive}}$$
$$= \frac{TP}{TP + FP}$$

Suppose that you take a test for a disease. The PPV captures the probability that you actually have the disease when you test positive.

Other metrics

Negative Predictive Value (NPV):

$$NPV = \frac{\#\text{ observations correctly classified as negative}}{\#\text{observations classified as negative}}$$
$$= \frac{TN}{TN + FN}$$

Suppose that you take a test for a disease. The NPV is the probability that you are not sick, given that the test comes up negative.

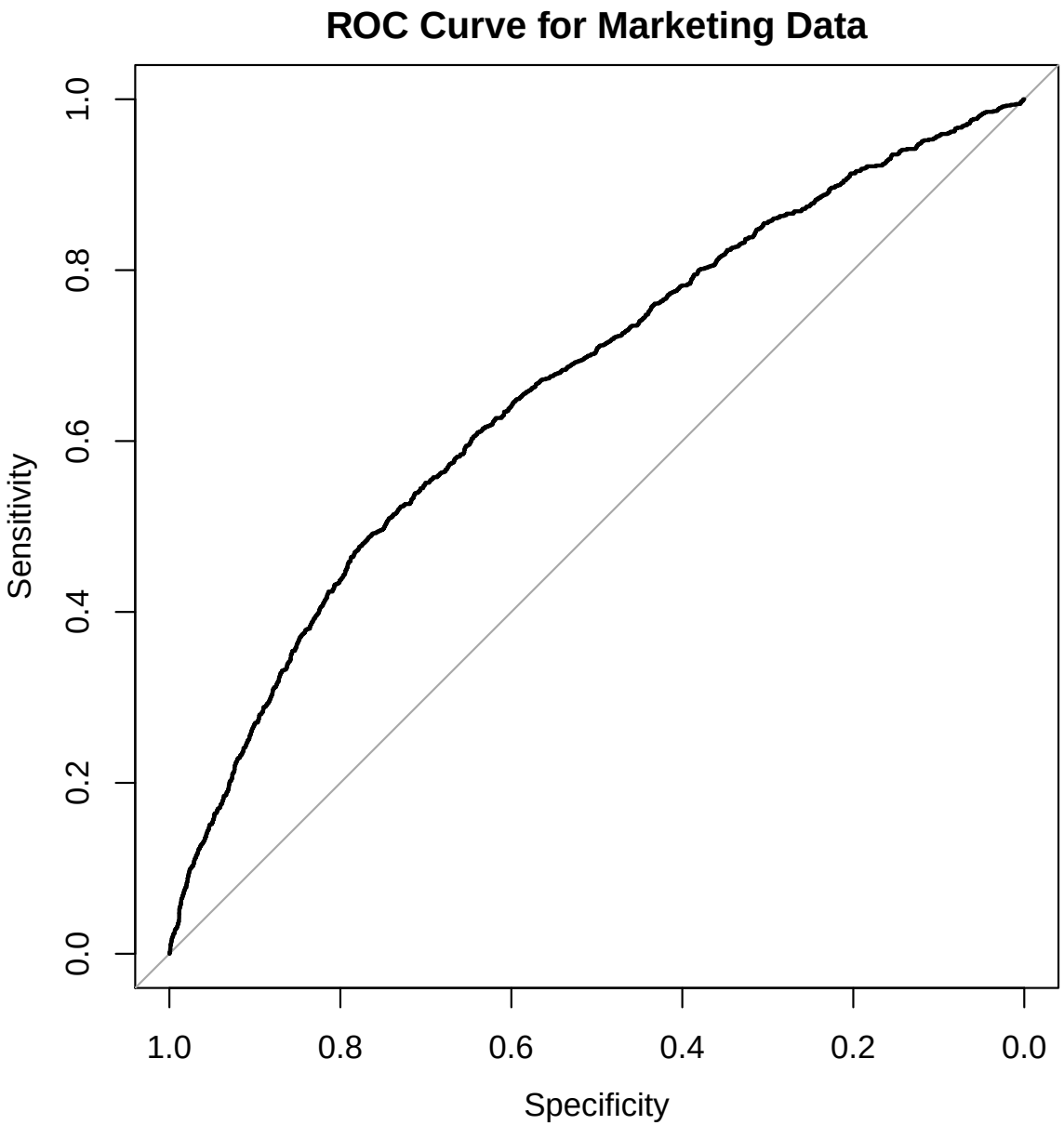
Changing threshold on $\mathbb{P}(Y = 1|X = x)$

To compare predictors based on their probability estimates (as opposed to their accuracies) we often use the ROC (Receiver operating characteristic) curve. As we change the threshold on $\mathbb{P}(Y = 1|X = x)$, we vary the size and quality of the set we reject.

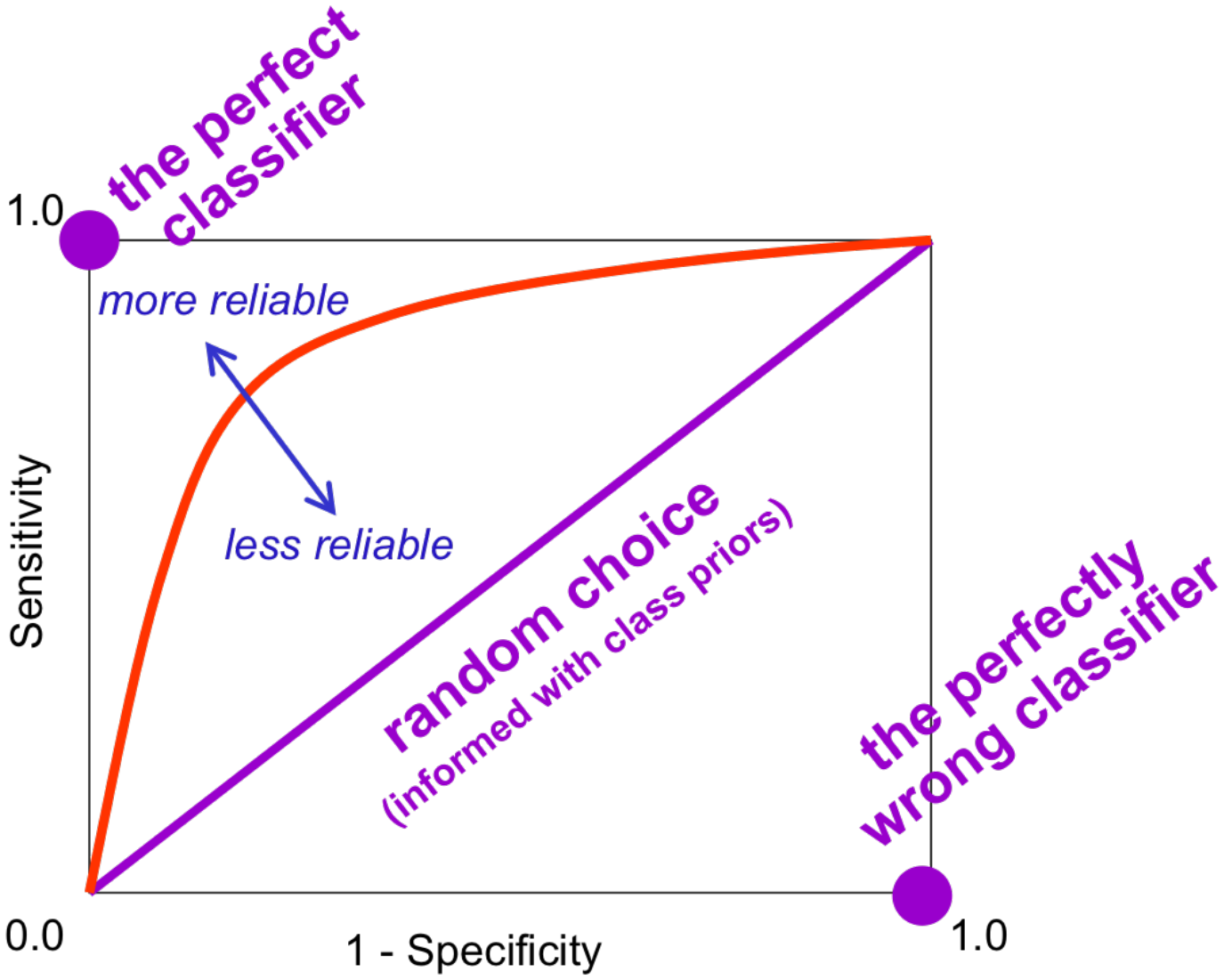
We can think about $\mathbb{P}(Y = 1|X = x)$ as a function for ranking our observations by likelihood of $Y = 1$.

It is convenient to look at the Sensitivity and Specificity of our classifier as the threshold changes. This corresponds to the ROC curve.

ROC Curve


ROC Curve for Marketing Data
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.8
0.6
0.4
0.2
0.0
Specificity

ROC Curve



How well does a random classifier do?

How well does a random classifier do?

- What is a random classifier?

ROC Curve

- A generally higher curve tends to be better, though you should keep tradeoffs in mind.
- The Area Under the Curve (AUC) is one way to measure this. A higher area tends to be more attractive.
- ROC curves will often cross each other and the AUC may not be satisfactory. You are really interested in the performance cutoff at a particular point on the curve

Linear Algebra Review

- A vector:
- A vector written in the standard basis:
- The length of a vector in the standard basis:

Linear Algebra Review

- A basis:

- Orthonormal matrices:

- Important properties of orthonormal matrices:

Eigendecomposition

- A square, real, symmetric matrix:

- Diagonalizing a square, real, symmetric matrix:

- Eigenvectors definition:

Eigendecomposition

- Why is the eigen-basis nice?

- Computing matrix-vector products:

- Computing matrix inverse/pseudo-inverse:

Singular Value Decomposition

- What if we have a non-square, non-symmetric matrix?