Performance Measures

Dan Lizotte

2017-03-02

Performance Measures for Classification

Why different performance measures?

To date, we have focussed on accuracy: How often is my classifier correct on new data?

Depending on how the classifier will be applied, however, other measures may be more appropriate.

Review: Error Rate / Accuracy

Compute the proportion that were correctly or incorrectly classified.

$$\text{Accuracy} = n^{-1} \sum_{i=1}^{n} 1(\hat{y}_i = y_i)$$

$$\text{ErrorRate} = n^{-1} \sum_{i=1}^{n} 1(\hat{y}_i \neq y_i)$$

Imbalanced classes

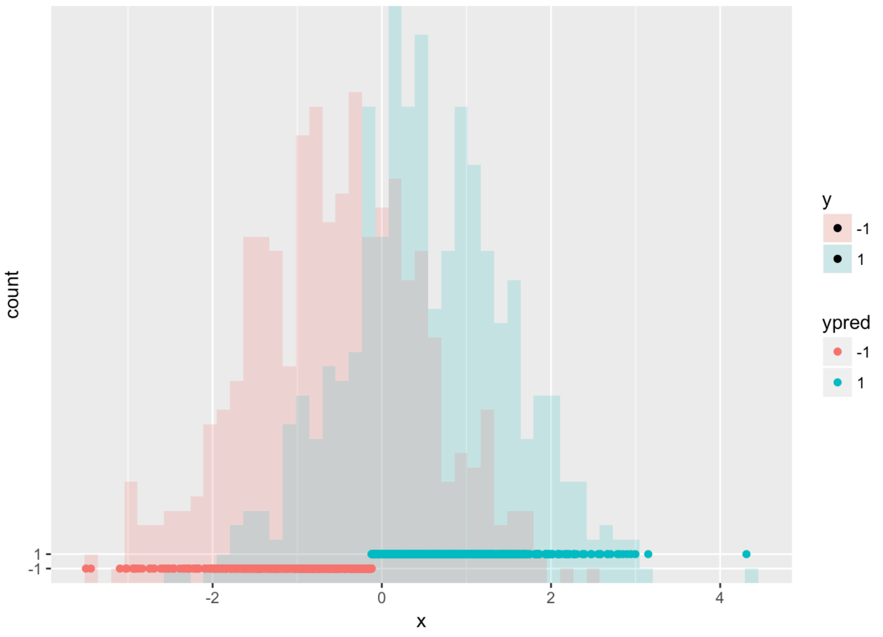
- Suppose in the true population, 95% are negative.
- Classifier that always outputs negative is 95% accurate. This is the **baseline** accuracy.
- Accuracy is not a useful measure.

Literature on learning from unbalanced classes:

http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5299216 (http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5299216) http://link.springer.com/article/10.1007%2Fs10115-013-0670-6 (http://link.springer.com/article/10.1007%2Fs10115-013-0670-6) http://www.computer.org/csdl/proceedings/icdm/2012/4905/00/4905a695-abs.html (http://www.computer.org/csdl/proceedings/icdm/2012/4905/00/4905a695-abs.html) http://www.computer.org/csdl/proceedings/icdm/2011/4408/00/4408a754-abs.html (http://www.computer.org/csdl/proceedings/icdm/2011/4408/00/4408a754-abs.html)

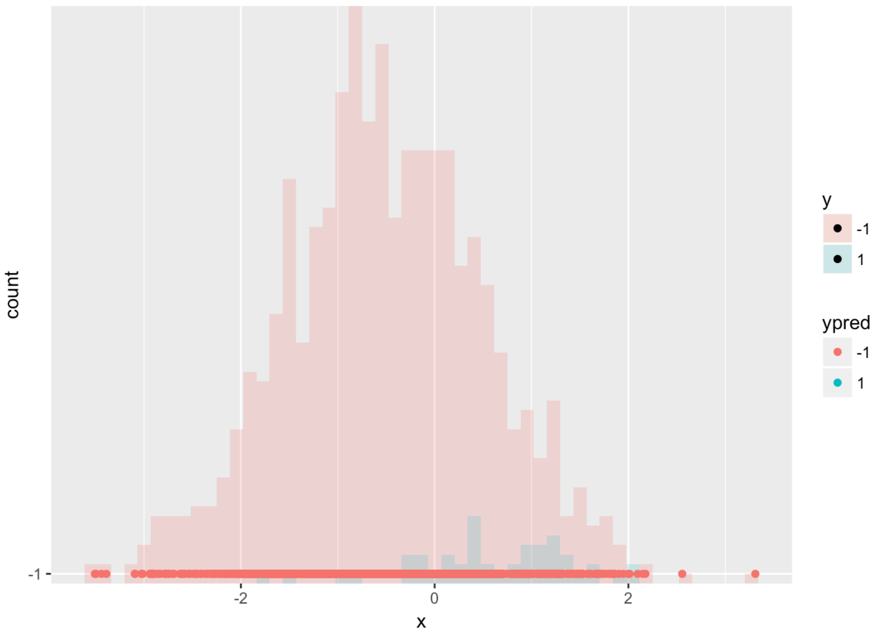
Example: 50% Positive, 50% Negative

```
npos <- 500; nneg <- 500; set.seed(1)
df <- rbind(data.frame(x=rnorm(npos,mupos), y=1),data.frame(x=rnorm(nneg,muneg),y=-1)); df$y <- as.factor(df$y)
sep <- tune(svm,y~x,data=df,ranges=list(gamma = 2^(-1:1), cost = 2^(2:4)))
df$ypred <- predict(sep$best.model)
ggplot(df,aes(x=x,fill=y)) + geom_histogram(alpha=0.2,position="identity",bins=51) + geom_point(aes(y=ypred,colou
r=ypred)) + scale_color_discrete(drop=FALSE)
```



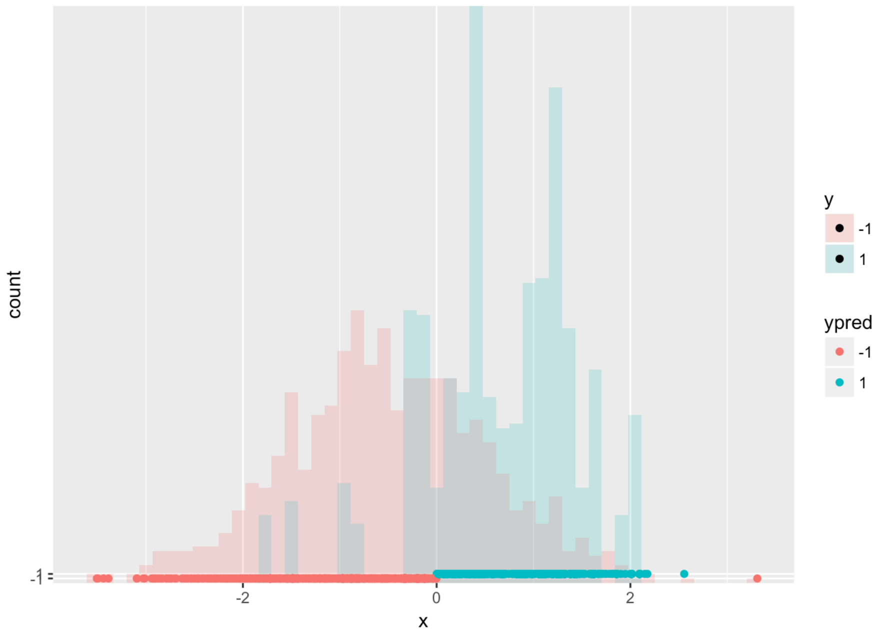
Example: 5% Positive, 95% Negative

```
npos <- 50; nneg <- 950; set.seed(1)
df <- rbind(data.frame(x=rnorm(npos,mupos), y=1),data.frame(x=rnorm(nneg,muneg),y=-1)); df$y <- as.factor(df$y)
rsep <- tune(svm,y~x,data=df,ranges=list(gamma = 2^(-1:1), cost = 2^(2:4)))
df$ypred <- predict(rsep$best.model);
ggplot(df,aes(x=x,fill=y)) + geom_histogram(alpha=0.2,position="identity",bins=51) + geom_point(aes(y=ypred,colou
r=ypred)) + scale_color_discrete(drop=FALSE)
```



Example: Upsampling

```
newneg <- df %>% filter(y == 1) %>% sample_n(900,replace=T); dfupsamp <- rbind(df,newneg)
upsep <- tune(svm,y~x,data=dfupsamp,ranges=list(gamma = 2^(-1:1), cost = 2^(2:4)))
dfupsamp$ypred <- predict(upsep$best.model);
ggplot(dfupsamp,aes(x=x,fill=y)) + geom_histogram(alpha=0.2,position="identity",bins=51) + geom_point(aes(y=ypred
,colour=ypred)) + scale_color_discrete(drop=FALSE)
```



Upsampling: Accuracy

```
df$upsampred <- predict(upsep$best.model,df)
mean(df$y == df$ypred)
```

```
## [1] 0.95
```

```
mean(df$y == df$upsampred)
```

```
## [1] 0.694
```

No upsampling: 95% Accuracy

Upsampled: 68% Accuracy

So why do you like the upsampled classifier better?

Definitions:
True/False Positives/Negatives

		True class	
	Total population (https://en.wikipedia.org/wiki/Statistical_population)	class positive	class negative
Predicted class	Predicted class positive	**True positive (https://en.wikipedia.org/wiki/True_positive)**	**False positive (https://en.wikipedia.org/wiki/False_positive)** (Type I error (https://en.wikipedia.org/wiki/Type_I_error))
	Predicted class negative	**False negative (https://en.wikipedia.org/wiki/False_negative)** (Type II error (https://en.wikipedia.org/wiki/Type_II_error))	**True negative (https://en.wikipedia.org/wiki/True_negative)**

Careful! A “false positive” is actually a *negative* and a “false negative” is actually a *positive*.

Precision and Recall, *F-measure*

Precision (https://en.wikipedia.org/wiki/Precision_(information_retrieval)) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted positive}}$ Recall (https://en.wikipedia.org/wiki/Recall_(information_retrieval)) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Class positive}}$

- In *Information Retrieval*, typically very few positives, many negatives. (E.g. billion webpages, dozen relevant to search query.) Focus is on correctly identifying positives.
- **Recall**: What proportion of the positives in the population do I correctly capture?
- **Precision**: What proportion of the instances I labeled positive are actually positive?

$$\text{F-measure} = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

https://en.wikipedia.org/wiki/F1_score (https://en.wikipedia.org/wiki/F1_score)

F-measure Example

$$\text{F-measure} = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

https://en.wikipedia.org/wiki/F1_score (https://en.wikipedia.org/wiki/F1_score)

For the “always predict -1” classifier, recall = 0, precision = 0, so
F-measure = 0.

For the classifier learned from up-sampled data,

```
prec <- sum(df$y == 1 & df$upsampred == 1) / sum(df$upsampred == 1)
recall <- sum(df$y == 1 & df$upsampred == 1) / sum(df$y == 1)
F1.upsamp <- 2 * prec*recall / (prec + recall)
print(F1.upsamp)
```

```
## [1] 0.2072539
```

NOTE that F-measure is not “symmetric”; it depends on the definition of the positive class. Typically used when positive class is rare but important e.g. information retrieval.

Sensitivity and Specificity, *Balanced Accuracy*

Sensitivity (https://en.wikipedia.org/wiki/Sensitivity_(tests)) = $\frac{\Sigma\text{ True positive}}{\Sigma\text{ Class positive}}$ Specificity (https://en.wikipedia.org/wiki/Specificity_(tests)) = $\frac{\Sigma\text{ True negative}}{\Sigma\text{ Class negative}}$

- **Sensitivity**: What proportion of the positives in the population do I correctly label?
- **Specificity**: What proportion of the negatives in the population do I correctly label?

$$\text{BalancedAccuracy} = \frac{1}{2}(\text{Sensitivity} + \text{Specificity})$$

https://en.wikipedia.org/wiki/Accuracy_and_precision#In_binary_classification (https://en.wikipedia.org/wiki/Accuracy_and_precision#In_binary_classification)

Note: Sensitivity is same as Recall.

Balanced accuracy Example

- **Sensitivity**: What proportion of the positives in the population do I correctly label?
- **Specificity**: What proportion of the negatives in the population do I correctly label?

$$\text{BalancedAccuracy} = \frac{1}{2}(\text{Sensitivity} + \text{Specificity})$$

For “always predict -1” classifier, sensitivity = 0, specificity = 1, balanced accuracy = 0.5.

For the classifier learned from up-sampled data,

```
sens <- sum(df$y == 1 & df$upsampred == 1) / sum(df$y == 1)
spec <- sum(df$y == -1 & df$upsampred == -1) / sum(df$y == -1)
bal.acc.upsamp <- 0.5*(sens + spec)
print(bal.acc.upsamp)
```

```
## [1] 0.7442105
```

Many Measures

https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers (https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers)

		True class	
	Total population (https://en.wikipedia.org/wiki/Statistical_population)	class positive	class negative
Predicted class	Predicted class positive	**True positive (https://en.wikipedia.org/wiki/True_positive)**	**False positive (https://en.wikipedia.org/wiki/False_p** (Type I error (https://en.wikipedia.org/wiki/Type_I_
	Predicted class negative	**False negative (https://en.wikipedia.org/wiki/False_negative)** (Type II error (https://en.wikipedia.org/wiki/Type_II_error))	**True negative (https://en.wikipedia.org/wiki/True_n**
	Accuracy (https://en.wikipedia.org/wiki/Accuracy_and_precision) (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	True positive rate (https://en.wikipedia.org/wiki/True_positive_rate) (TPR), Sensitivity (https://en.wikipedia.org/wiki/Sensitivity_(tests)), Recall (https://en.wikipedia.org/wiki/Recall_(information_retrieval)) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Class positive}}$	False positive rate (https://en.wikipedia.org/wiki/False_posi (FPR), Fall-out (https://en.wikipedia.org/wiki/Information = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Class negative}}$
		False negative rate (https://en.wikipedia.org/wiki/False_negative_rate) (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Class positive}}$	True negative rate (https://en.wikipedia.org/wiki/True_negat (TNR), Specificity (https://en.wikipedia.org/wiki/Specificity (SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Class negative}}$

Cost sensitivity

Sensitivity (https://en.wikipedia.org/wiki/Sensitivity_(tests)) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Class positive}}$ Specificity (https://en.wikipedia.org/wiki/Specificity_(tests)) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Class negative}}$

Recall:

$$\text{BalancedAccuracy} = \frac{1}{2}(\text{Sensitivity} + \text{Specificity})$$

What if e.g. false positives are more costly than false negatives?

Let P and N the proportions of positives and negatives in the population.

$$\text{FNRate} = (1 - \text{Sensitivity}), \text{FPRate} = (1 - \text{Specificity})$$

$$\text{NormExpectedCost} = c_{\text{FP}} \cdot \text{FPRate} \cdot \mathrm{P} + c_{\text{FN}} \cdot \text{FNRate} \cdot \mathrm{N}$$

http://www.csi.uottawa.ca/~cdrummon/pubs/pakdd08.pdf (http://www.csi.uottawa.ca/~cdrummon/pubs/pakdd08.pdf)

Receiver operating characteristic (ROC)

- Suppose classifier can *rank* inputs according to “how positive” they appear to be.
- E.g., can use probability from Logistic Regression, or $w^\mathsf{T} x + b$ for SVM.
- By adjusting the “threshold” value for deciding an instance is positive, we can obtain different false positive rates. Low threshold gives higher false positives (but higher true negatives), high threshold gives lower false positives (but higher false negatives.)
- ROC curve: Try all possible cutoffs, plot FPR on x-axis, TPR on y-axis.

https://en.wikipedia.org/wiki/Receiver_operating_characteristic (https://en.wikipedia.org/wiki/Receiver_operating_characteristic)

Reading an ROC Curve

Think: “If I fix FPR at 0.4, what is my TPR?”

Obviously, higher is better. Random guessing gives an ROC curve along $y = x$.

If the **area under the curve (AUC)** is 1, we have a perfect classifier. AUC of 0.5 is pretty bad.

Very common measure of classifier performance, especially when classes are imbalanced.

ROC example

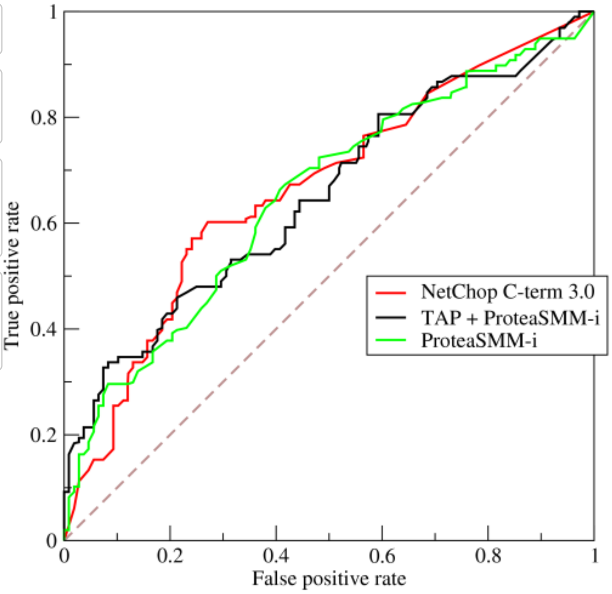
```
library(ROCR)
```

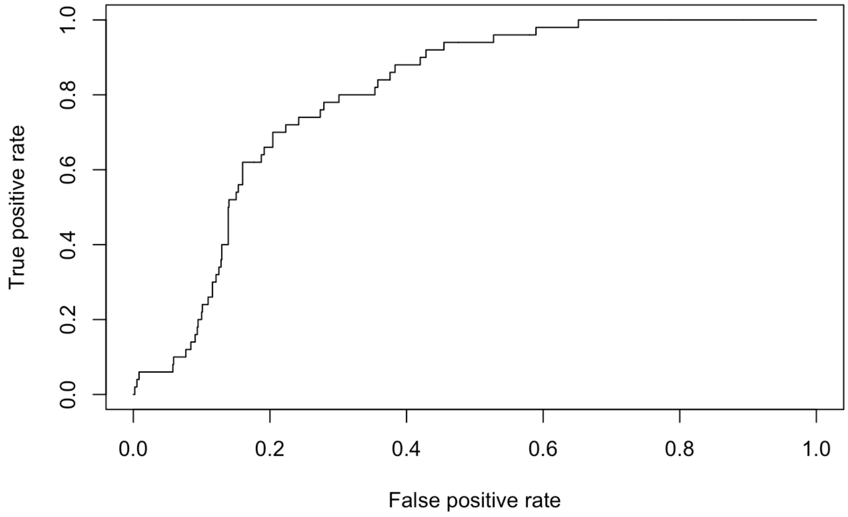
```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
preds <- attr(predict(upsep$best.model,df,decision.values = TRUE),"decision.values")
plot(performance(prediction(preds,df$y),"tpr","fpr"))
```





AUROC, C-statistic

- The Area Under the Reciever Operating Characteristic Curve is also called the C-statistic
- Also the statistic for the Wilcoxon-Mann-Whitney hypothesis test of equal distributions

Big picture: Optimizing classifiers

If we care about all these measures, why do we optimize misclassification rate, or margin, or likelihood?

- Computational tractability
- Classifier learned the way we described often perform well measures presented here

- **However**
  - There are methods for learning e.g. SVMs by optimizing ROC
  - Cost-sensitive learning is also widespread
  - Methods are evolving; a quick google scholar search is a good idea.

Performance Measures for Regression

Mean Errors

$$\mathrm{MSE} = n^{-1} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

$$\mathrm{RMSE} = \sqrt{n^{-1} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2}$$

$$\mathrm{MAE} = n^{-1} \sum_{i=1}^{n} |\hat{y}_i - y_i|$$

I find MAE easier to interpret. (How far am I from the correct value, on average?) RMSE is at least in the same units as the y.

Mean Relative Error

$$\mathrm{MRE} = n^{-1} \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{|y_i|}$$

Scales error according to magnitude of true y. E.g., if MRE=0.2, then regression is wrong by 20% of the value of y, on average.

If this is appropriate for your problem then linear regression, which assumes additive error, may not be appropriate. Options include using a different model or regression on $\log y$ rather than on y.

https://en.wikipedia.org/wiki/Approximation_error#Formal_Definition (https://en.wikipedia.org/wiki/Approximation_error#Formal_Definition)