Error and Noise

Error Measures

An error measure quantifies the performance of an $h \in \mathcal{H}$, that is, its agreement with the target function f.

$$Error = E(h, f)$$

The target function is unknown and we only have samples from it (our data set, $\mathcal{D}$), so we use a pointwise approximation.
Classification error is

$$e(h(\vec{x}), f(\vec{x})) = [\![h(\vec{x}) \neq f(\vec{x})]\!]$$

for some $\vec{x}$, where $[\![\cdot]\!]$ evaluates to 1 if argument is true, and to 0 if it is false.

Error Rate and Accuracy Rate

Given the previous pointwise definition of error, error rate within a data set $\mathcal{D}$ can be defined as

$$E(h) = \frac{1}{N} \sum_{n=1}^{N} [\![h(\vec{x}) \neq f(\vec{x})]\!]$$

In other words, it's the proportion of points in $\mathcal{D}$ that are misclassified by h. If you turn the inequality above into an equality, you have *accuracy*, that is, accuracy $= 1 - E$. Some prefer to think in terms of accuracy.

Georgia Tech

Training Error and Test Error

The E we just defined is the error of our h in $\mathcal{D}$, a set of samples from $\mathcal{X}$. The book refers to this quantity as *in-sample* error, or E_{in}.

- With our data set $\mathcal{D}$ we can only deal with E_{in}.
- What we really care about is E_{out} – how will our classifier perform on any possible unseen $\vec{x}$ from $\mathcal{X}$.

So in practice we separate our data set $\mathcal{D}$ into a training set and a test set.

- E_{train} is the error rate on our training set.
- E_{test} is the error rate on our test set.

We use E_{test} as an estimate of E_{out}. For this estimate to be meaningful we must observe the most critical rule in practical machine learning

You must not use any data from the test set during training.

Cost

E can be thought of as the *cost* of using h instead of f (if you knew f you'd just use f). But the error measure we just defined might not be enough. Consider the case of identification by fingerprint [1]:

$$\rightarrow f \rightarrow \begin{cases} +1 \text{you} \\ -1 \text{not you} \end{cases}$$

Is the cost of correctly identifying a person the same for all applications?

[1] Fingerprint image by Cyrillic at the English language Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=3335963

Kinds of Error

		f	
		+1	-1
h	+1	no error false	false positive
	-1	negative	no error

Consider the following two kinds of applications:

- Customer identification for a supermarket discount program
- Identification for authorization to enter CIA building

Is the cost for each kind of error the same?

Cost Matrix

We can capture the relative cost of each kind of error in a cost matrix.

		f: $+1$	f: -1
h	$+1$	0	1
	-1	10	0

Supermarket

		f: $+1$	f: -1
h	$+1$	0	1000
	-1	1	0

CIA

- Accidentally letting someone into the CIA building is 1000 times worse than accidentally rejecting someone
- A learning algorithm using a cost-weighted error function will minimize the right kind of error

Additional Error Metrics

Our earlier error function didn't distinguish between different kinds of errors – only misclassifications.

Let

- TP be the number of true positive predictions,
- TN be the number of true negative predictions,
- FP be the number of false positive predictions, and
- FN be the number of false negative predictions.

Then . . .

Precision, Recall, F-measure

- $Precision = \frac{TP}{TP+FP}$
  - If high, a positive prediction is likely correct (good for CIA entry)
  - Also called "hit rate"
- $Recall = \frac{TP}{TP+FN}$
  - If high, missed few positives but maybe had some false positives
  - Also called "false alarm rate"
  - Good for cancer diagnosis - better to scare someone than to miss an actual cancer
- $F1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$
  - If high, good precision *and* recall summarized in a single metric

Georgia Tech

Confusion Matrix

We can calculate the error metrics from a *confusion matrix*. A confusion matrix lists the counts of the different kinds of errors. We've switched the positions of the true function, f, and our learned hypothesis, h, to match the output of most machine learning libraries.

Let's say we run a simple linear discriminant analysis on the Wisconsin Breast Cancer Diagnostic data set and get the following confusoin matrix:

		h	
		+1	-1
truth	+1	48	7
	-1	0	88

Evaluating a Model using Precision, Recall, and F-measure

		h	
		+1	-1
truth	+1	48	7
	-1	0	88

Using these values,

- $Precision = \frac{TP}{TP+FP} = \frac{48}{48+0} = 1.0$
- $Recall = \frac{TP}{TP+FN} = \frac{48}{48+7} = 0.87$
- $F1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = 2 \cdot \frac{1.0 \cdot 0.87}{1.0+0.87} = 0.93$

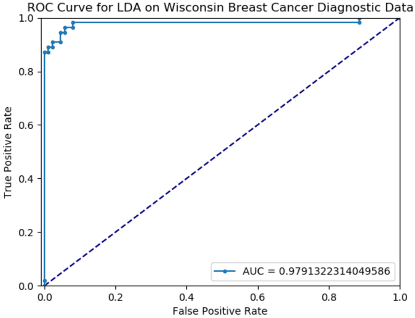
Forget, for a moment, that this model was evaluated on a breast cancer detection data set.

- For what kinds of applications would this be a good result?
- For what kinds of applications would this be a bad result?

BTW, the simple accuracy rate for this classifier would be $\frac{48+88}{48+88+7} = 0.95$

ROC Curves

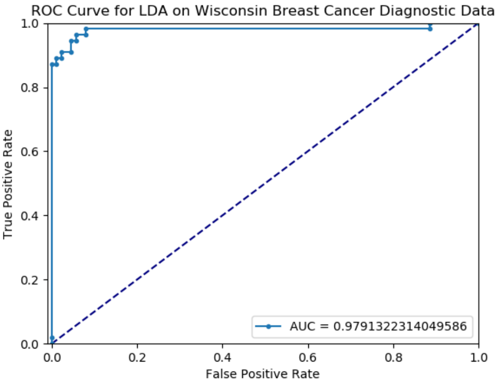
A *receiver operating characteristics curve*[2], or *ROC* curve, plots the tp-rate versus the fp-rate to show the tradeoffs of a particular model on a particular data set. Here's a ROC curve for a linear discriminant analysis model on the breast cancer data:



How do we interpret a ROC curve? . . .

[2] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.646.2144

Interpreting ROC Curves



- A better ROC curve will "hug" the upper-left corner
- Area under the curve, or AUC, is a good single-number measure of overall performance.

Calculating the previous metrics is straightforward, but plotting a ROC curve requires internal data used by a model. Let's see this in code ...

Closing Thoughts

- Error (aka cost, aka loss) is the difference between the true target function and our hypothesis.
- We estimate the error with pointwise evaluations and a learning algorithm may optimize this directly.
- We train a model using a training set and evaluate it (estimate true error) by calculating error on a test set.
- There are two kinds of errors: false positives and false negatives.
  - We can characterize the cost of the different kinds of errors for a particular application.
- Simple error or accuracy rate is a poor metric.
- We can count the true positives, false positives, true negatives and false negatives in a classifiers predicitons on the test set.
  - Using these counts we can calculate more fine-grained metrics for evaluating a classifier.
  - A ROC curve can give us a good general view of a classifier's general performance and tradeoffs.

and finally . . .

The Golden Rule

We must *never* use test data for training.

- Ideally we'd have a test set unavailable to us (like in competitions).
- In practice we split a data set into training and test sets during model development

In case you missed it,

WE MUST NEVER USE TEST DATA FOR TRAINING.

Georgia Tech