Advanced Evaluation, Imbalanced Data

Instructor: Jessica Wu -- Harvey Mudd College

The instructor gratefully acknowledges Eric Eaton (UPenn), David Kauchak (Pomona), Tommi Jaakola (MIT) and the many others who made their course materials freely available online.

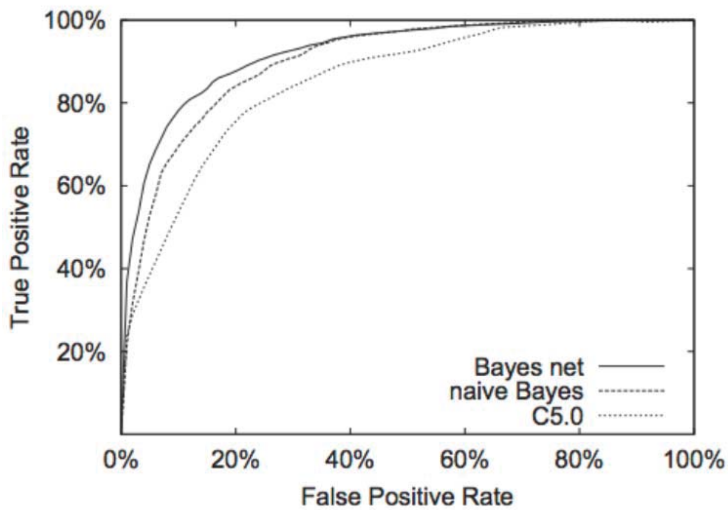
Advanced Evaluation Metrics

Learning Goals

- Describe metrics for evaluating performance
  - AUROC, precision, recall, F_1-score

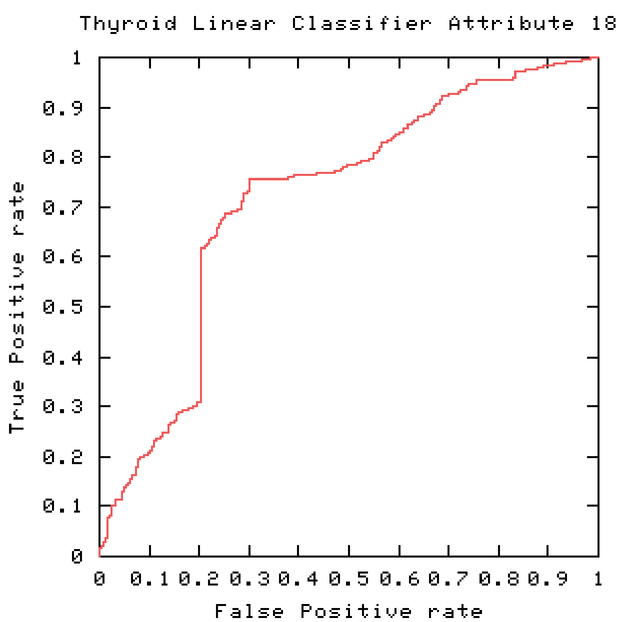
ROC Curves

Obtain curve by varying threshold on confidence of instance being positive



Based on slide by David Page
[Image source: Bockhorst et al., *Bioinformatics* 2003]

ROC Curves and Misclassification Costs



Assume balanced classes

Identify

a) best operating point when cost of misclassifying positives and negatives is equal

b) best operating point when FP costs 10x FN

c) best operating point when FN costs 10x FP

Based on slide by David Page
[Image source: http://www0.cs.ucl.ac.uk/staff/ucacbbl/roc/]

Algorithm for Creating ROC Curve

- Sort test set predictions according to confidence that instance is positive
- Step through sorted list from high to low confidence
  - locate *threshold* between instances with opposite classes (keeping instances with same confidence value on same side of threshold)
  - compute TPR, FPR for instances above threshold
  - output (FPR, TPR) coordinate

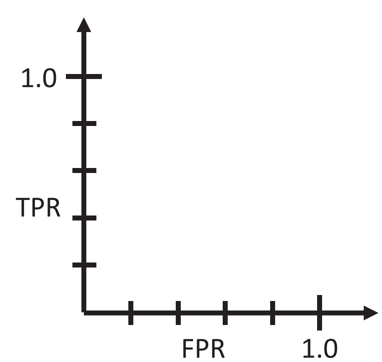
Based on slide by David Page

Plotting an ROC Curve

		predicted class	
		yes	no
actual class	yes	TP	FN
	no	FP	TN

TPR = TP / (TP + FN)
FPR = FP / (TN + FP)

instance	confidence in positive	correct class
9	0.99	+
7	0.98	+
1	0.72	-
2	0.70	+
6	0.65	+
10	0.51	-
3	0.39	-
5	0.24	+
4	0.11	-
8	0.01	-



Based on slide by David Page

But wait…

Does low FPR (high specificity) indicate that most positive predictions (predictions with confidence > some threshold) are correct?

suppose TPR = 0.9, FPR = 0.01

fraction of instances that are positive	fraction of positive predictions that are correct
0.5	0.989
0.1	0.909
0.01	0.476
0.001	0.083

		predicted class	
		yes	no
actual class	yes	TP	FN
	no	FP	TN

TPR = TP / (TP + FN)
FPR = FP / (TN + FP)
PR (positive rate) = (TP + FN) / total
NR (negative rate) = (TN + FP) / total

fraction of positive predictions
that are correct
= TP / (TP + FP)
= TPR * PR / (TPR * PR + FPR * NR)

Based on slide by David Page

Confusion Matrix

Given dataset of P positive instances and N negative instances:

		predicted class	
		Yes	No
actual class	Yes	TP	FN
	No	FP	TN

$$\text{accuracy} = \frac{TP + TN}{P + N}$$

Imagine a classifier that identifies presence of disease

$$\text{sensitivity} = \frac{TP}{TP + FN}$$

(true positive rate) = probability of positive test given person has disease

$$\text{specificity} = \frac{TN}{TN + FP}$$

(true negative rate) = probability of negative test given person does not have disease

$$\text{precision} = \frac{TP}{TP + FP}$$

(positive predictive value) = probability that person has disease given positive test

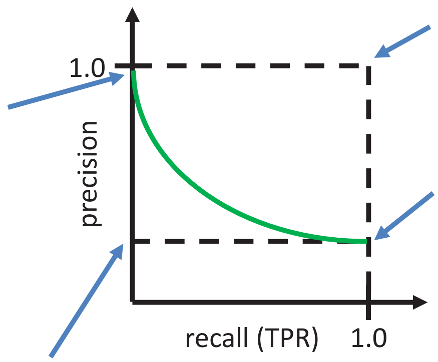
$$\text{recall} = \frac{TP}{TP + FN}$$

(true positive rate) = probability of positive test given person has disease

Based on slide by Eric Eaton

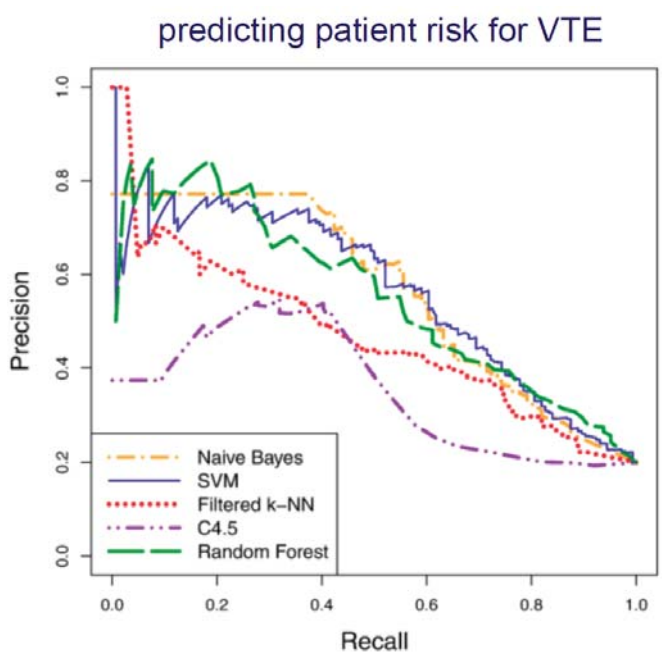
Precision-Recall Curves

Plots precision vs recall as you vary threshold on confidence of instance being positive



Based on slide by David Page

Precision-Recall Curve Example



Based on slide by David Page

[Image source: Kawaler et al., *Proc of AMIA Annual Symposium*, 2012]

Best Operating Point

Compromise between precision and recall
(harmonic mean)

$$F_\beta = \frac{(1+\beta^2)\cdot \text{precision}\cdot\text{recall}}{\beta^2\cdot\text{precision}+\text{recall}} = \frac{1}{\alpha\frac{1}{\text{precision}} + (1-\alpha)\frac{1}{\text{recall}}}$$

$$\alpha = \frac{1}{1+\beta^2}$$

F_1 measure most common
(α = 0.5, β = 1, precision and recall weighted equally)

$$F_1 = 2\cdot\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$$

Metrics Exercise

		predicted class	
		yes	no
actual class	yes	100	5
	no	10	50

sensitivity?

specificity?

precision?

recall?

F_1-score?

Why Harmonic Mean?

- Punishes *extreme* values more

Example
- – dataset: infinite examples of negative class; one example of positive class
- – classifier: (trivial) always predict positive
- – then: precision = recall =
  arithmetic mean = harmonic mean (F_1) =

$\Rightarrow$ for a high F_1, need *both* high precision and recall

- Mathematically correct

harmonic mean = reciprocal of arithmetic mean of reciprocals

$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

F_1-score takes averages over the same denominator

Comments on ROC and PR curves

Both
- allow predictive performance to be assessed at various levels of confidence
- assume binary classification tasks
- sometimes summarized by calculating *area under the curve* (AUROC, AUPR)

ROC curves
- insensitive to changes in class distribution
  (ROC curve does not change if proportion of positive and negative instances in test set are varied)
- can identify optimal classification thresholds for tasks with differential misclassification costs

PR curves
- show fraction of predictions that are false positives
- well-suited for tasks with lots of negative instances

Based on slide by David Page

A Word of Caution

Consider binary classifiers A, B, C

		A	.	B	.	C	.
		1	0	1	0	1	0
Predictions	1	0.9	0.1	0.8	0	0.78	0
	0	0	0	0.1	0.1	0.12	0.1

- Clearly A is useless since it always predicts 1
- B is slightly better than C
  - less probability mass wasted on off-diagonals
- But, here are the performance metrics

Metric	A	B	C
Accuracy	0.9	0.9	0.88
Precision	0.9	1.0	1.0
Recall	1.0	0.888	0.8667
F-score	0.947	0.941	0.9286

Based on slide by Kevin Murphy

Imbalanced Data

Learning Goals

- Describe approaches for handling imbalanced data and the trade-offs of each

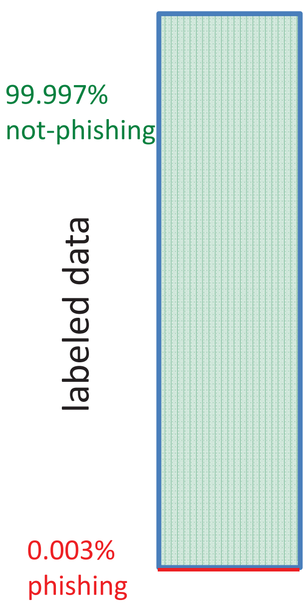
Setup

1. for 1 hour, Google collects 1M e-mails randomly
2. they pay people to label them as "phishing" or "not-phishing"
3. they give the data to you to learn to classify e-mails as phishing or not
4. you, having taken ML, try out a few of your favorite classifiers
5. you achieve an accuracy of 99.997%

Should you be happy?

Based on slide by David Kauchak

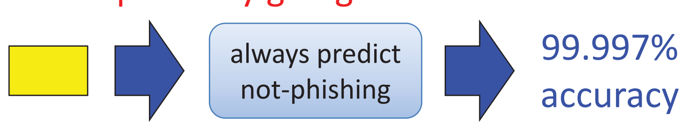
Imbalanced Data



The phishing problem is what is called an **imbalanced data** problem

- occurs where there is large discrepancy between number of examples with each class label
- e.g. for our 1M example dataset, only ~30 would actually represent phishing e-mails

What is probably going on with our classifier?



Why does the classifier learn this?

- Many classifiers are designed to optimize error/accuracy
- This tends to bias performance towards majority class
- *Anytime* there is imbalance in the data, this can happen
- It is particularly pronounced, though, when imbalance is more pronounced

Based on slide by David Kauchak

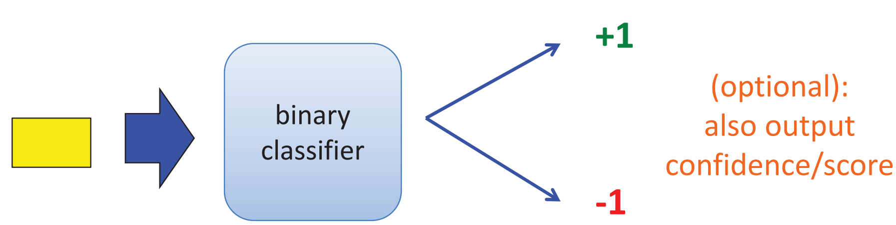
Imbalanced Problem Domains

Besides phishing (and spam), what are some other imbalanced problems domains?

- Medical diagnosis
- Predicting faults/failures (e.g. hard-drive failures, mechanical failures, etc.)
- Predicting rare events (e.g. earthquakes)
- Detecting fraud (credit card transactions, internet traffic)

Black-Box Approach

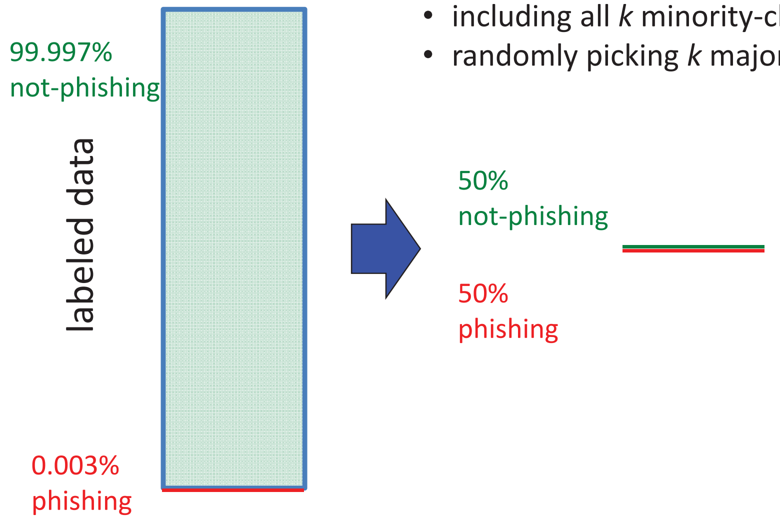
Abstraction: We have generic binary classifier. How can we use it to solve our new problem?



Can we do some pre-processing/post-processing of our data to allow us to still use our binary classifiers?

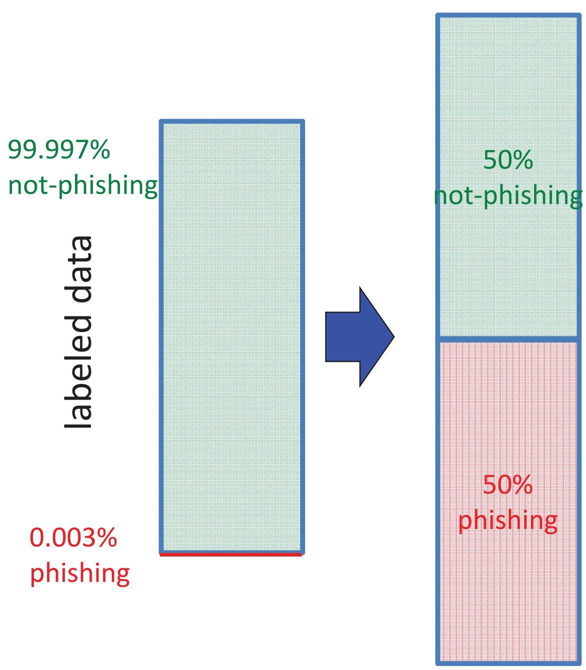
Idea 1: Subsampling

Create a new training data set by
- including all *k* minority-class examples
- randomly picking *k* majority-class examples



Based on slide by David Kauchak

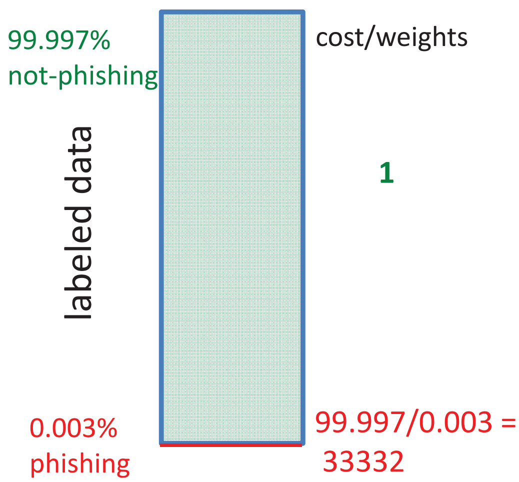
Idea 2: Oversampling



Create a new training data set by
- including all *m* majority-class examples
- including *m* minority-class examples:
  - repeat each example fixed number of times, or
  - sample with replacement

Based on slide by David Kauchak

Idea 2b: Weighted Examples



Add costs/weights to training set

- majority-class examples get weight 1
- minority-class examples get much larger weight

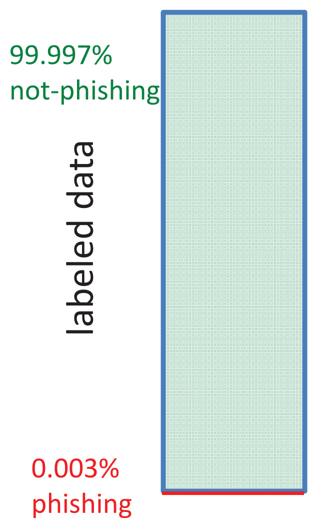
Change learning algorithm to optimize weighted error

Idea 3: Optimize Different Metric

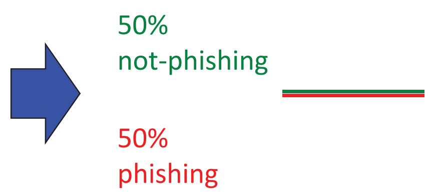
Train classifiers that try and optimize F_1 or AUC or …

come up with another learning algorithm designed specifically for imbalanced problems

Idea 1: Subsampling



Create a new training data set by
- including all *k* minority-class examples
- randomly picking *k* majority-class examples



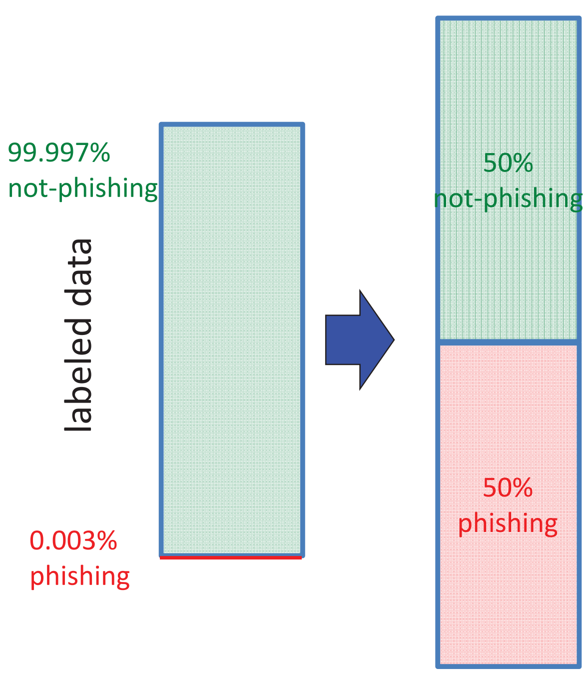
pros / cons?

Pros
- Easy to implement
- More efficient training (smaller training set)
- For some domains, can work very well

Cons
- Throwing away lots of data/information

Based on slide by David Kauchak

Idea 2: Oversampling



Create a new training data set by
- including all *m* majority-class examples
- include *m* minority-class examples:
  - repeat each example fixed number of times, or
  - sample with replacement

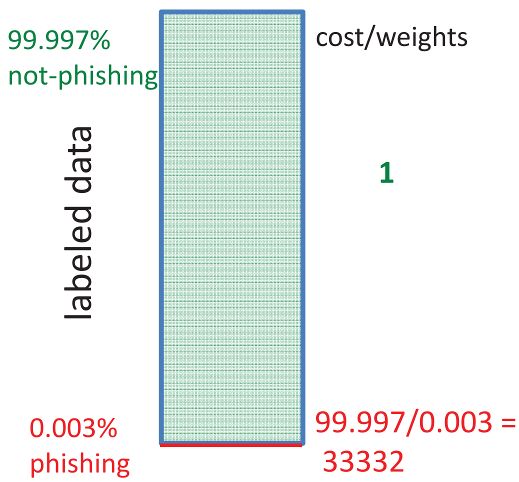
pros / cons?

Pros
- Easy to implement
- Utilizes all training data
- Tends to perform well in broader set of circumstances than subsampling

Cons
- Computationally expensive to train classifier

Based on slide by David Kauchak

Idea 2b: Weighted Examples



Add costs/weights to training set
- majority-class examples get weight 1
- minority-class examples get much larger weight

Change learning algorithm to optimize weighted error

pros / cons?

Pros
- Achieves effect of oversampling without computational cost
- Utilizes all training data
- Tends to perform well in broader set of circumstances

Cons
- Requires classifier that can deal with weights

Based on slide by David Kauchak

Idea 3: Optimize Different Metric

Train classifiers that try and optimize F_1 or AUC or …

come up with another learning algorithm designed specifically for imbalanced problems

pros/cons?

- Not all classifiers amenable to optimizing F_1 or AUC
- Do not want to reinvent the wheel – that said, there are a number of approaches specifically developed to handle imbalanced problems

Based on slide by David Kauchak