Evaluating Machine Learning Methods

CSE 4334 / 5334 Data Mining
Spring 2019

Won Hwa Kim

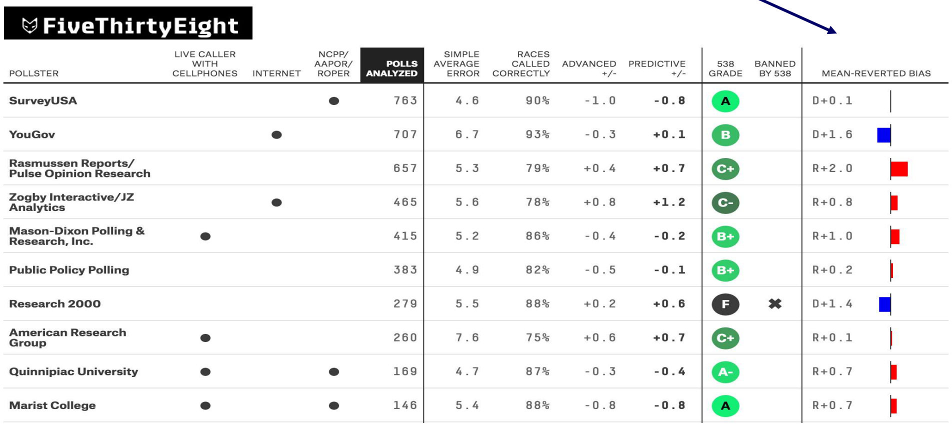
Part of the contents borrowed from Prof. Mark Craven / Prof. David Page Jr. at UW-Madison

Bias of an estimator

θ true value of parameter of interest (e.g. model accuracy)
$\hat{\theta}$ estimator of parameter of interest (e.g. test set accuracy)

$$\mathrm{Bias}[\hat{\theta}] = \mathrm{E}[\hat{\theta}] - \theta$$

e.g. polling methodologies often have an inherent bias

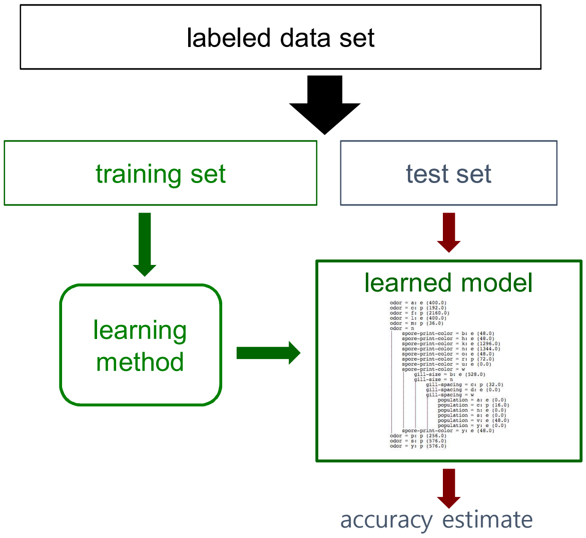


FiveThirtyEight

POLLSTER	LIVE CALLER WITH CELLPHONES	INTERNET	NCPP/ AAPOR/ ROPER	POLLS ANALYZED	SIMPLE AVERAGE ERROR	RACES CALLED CORRECTLY	ADVANCED +/-	PREDICTIVE +/-	538 GRADE	BANNED BY 538	MEAN-REVERTED BIAS
SurveyUSA			●	763	4.6	90%	-1.0	**-0.8**	A		D+0.1
YouGov		●		707	6.7	93%	-0.3	**+0.1**	B		D+1.6
Rasmussen Reports/ Pulse Opinion Research				657	5.3	79%	+0.4	**+0.7**	C+		R+2.0
Zogby Interactive/JZ Analytics		●		465	5.6	78%	+0.8	**+1.2**	C-		R+0.8
Mason-Dixon Polling & Research, Inc.	●			415	5.2	86%	-0.4	**-0.2**	B+		R+1.0
Public Policy Polling				383	4.9	82%	-0.5	**-0.1**	B+		R+0.2
Research 2000				279	5.5	88%	+0.2	**+0.6**	F	✖	D+1.4
American Research Group	●			260	7.6	75%	+0.6	**+0.7**	C+		R+0.1
Quinnipiac University	●		●	169	4.7	87%	-0.3	**-0.4**	A-		R+0.7
Marist College	●		●	146	5.4	88%	-0.8	**-0.8**	A		R+0.7

Test sets

How can we get an unbiased estimate of the accuracy of a learned model?



```
odor = a: e (400.0)
odor = c: p (192.0)
odor = f: p (2160.0)
odor = l: e (400.0)
odor = m: p (36.0)
odor = n
|   spore-print-color = b: e (48.0)
|   spore-print-color = h: e (48.0)
|   spore-print-color = k: e (1296.0)
|   spore-print-color = n: e (1344.0)
|   spore-print-color = o: e (48.0)
|   spore-print-color = r: p (72.0)
|   spore-print-color = u: e (0.0)
|   spore-print-color = w
|   |   gill-size = b: e (528.0)
|   |   gill-size = n
|   |   |   gill-spacing = c: p (32.0)
|   |   |   gill-spacing = d: e (0.0)
|   |   |   gill-spacing = w
|   |   |   |   population = a: e (0.0)
|   |   |   |   population = c: p (16.0)
|   |   |   |   population = n: e (0.0)
|   |   |   |   population = s: e (0.0)
|   |   |   |   population = v: e (48.0)
|   |   |   |   population = y: e (0.0)
|   spore-print-color = y: e (48.0)
odor = p: p (256.0)
odor = s: p (576.0)
odor = y: p (576.0)
```

Test sets

How can we get an unbiased estimate of the accuracy of a learned model?

- when learning a model, you should pretend that you don't have the test data yet (it is "in the mail")*
- if the test-set labels influence the learned model in any way, accuracy estimates will be biased

* In some applications it is reasonable to assume that you have access to the feature vector (i.e. x) but not the y part of each test instance.

Learning curves

How does the accuracy of a learning method change as a function of the training-set size?

this can be assessed by plotting *learning curves*

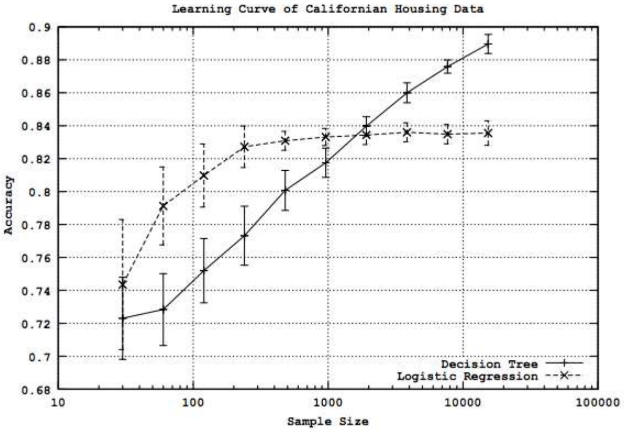
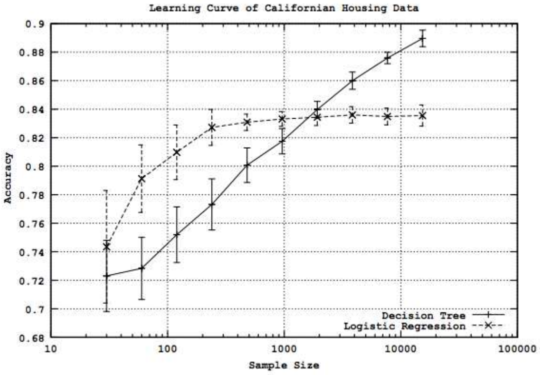


Figure from Perlich et al. *Journal of Machine Learning Research*, 2003

Learning curves

given training/test set partition

- for each sample size s on learning curve
  - (optionally) repeat n times
    - randomly select s instances from training set
    - learn model
    - evaluate model on test set to determine accuracy a
    - plot (s, a)
    - or (s, avg. accuracy and error bars)



Limitations of using a single training/test partition

- we may not have enough data to make sufficiently large training and test sets
  - a larger test set gives us more reliable estimate of accuracy (i.e. a lower variance estimate)
  - but… a larger training set will be more representative of how much data we actually have for learning process
- a single training set doesn't tell us how sensitive accuracy is to a particular training sample

Using multiple training/test partitions

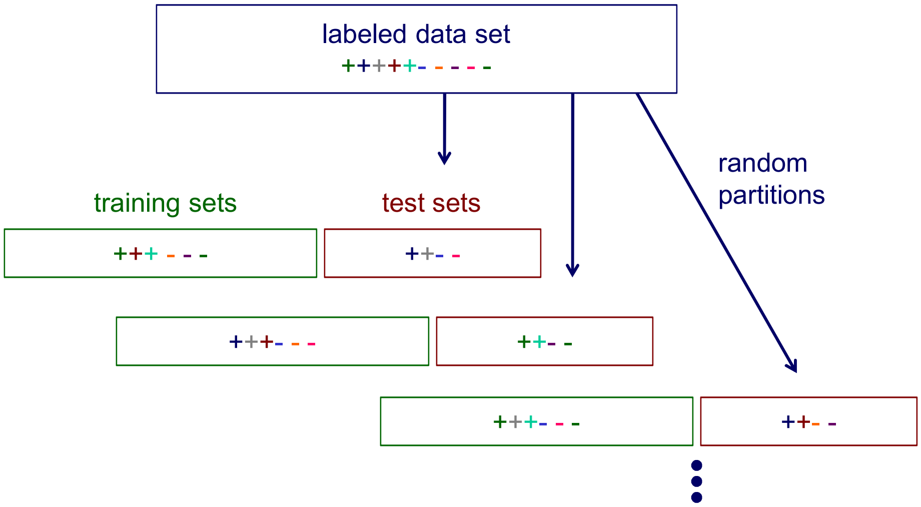
- Two general approaches…

  - random resampling

  - cross validation

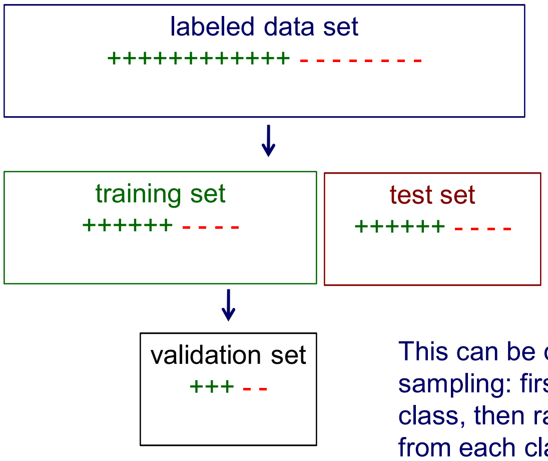
Random Resampling

We can address the second issue by repeatedly randomly partitioning the available data into training and set sets.



Stratified Resampling

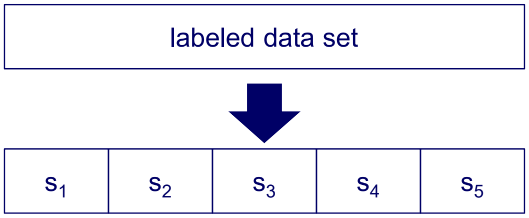
When randomly selecting training or validation sets, we may want to ensure that class proportions are maintained in each selected set



This can be done via stratified sampling: first stratify instances by class, then randomly select instances from each class proportionally.

Cross validation

partition data
into *n* subsamples



iteratively leave one subsample out for the test set, train on the rest

iteration	train on	test on
1	s_2 s_3 s_4 s_5	s_1
2	s_1 s_3 s_4 s_5	s_2
3	s_1 s_2 s_4 s_5	s_3
4	s_1 s_2 s_3 s_5	s_4
5	s_1 s_2 s_3 s_4	s_5

Cross validation example

Suppose we have 100 instances, and we want to estimate accuracy with cross validation

iteration	train on	test on	correct
1	s_2 s_3 s_4 s_5	s_1	11 / 20
2	s_1 s_3 s_4 s_5	s_2	17 / 20
3	s_1 s_2 s_4 s_5	s_3	16 / 20
4	s_1 s_2 s_3 s_5	s_4	13 / 20
5	s_1 s_2 s_3 s_4	s_5	16 / 20

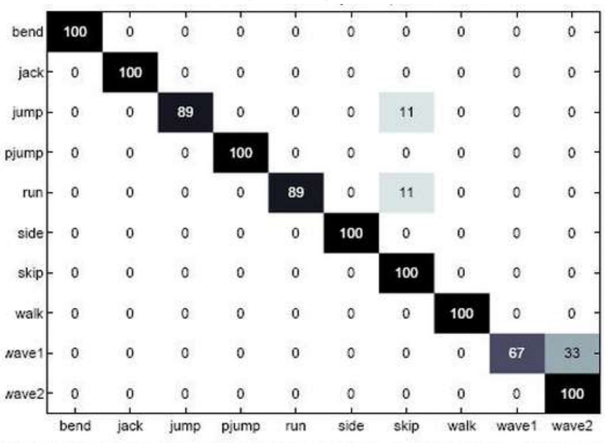
accuracy = 73/100 = 73%

Cross validation

- 10-fold cross validation is common, but smaller values of *n* are often used when learning takes a lot of time
- in *leave-one-out* cross validation, *n* = # instances
- in *stratified* cross validation, stratified sampling is used when partitioning the data
- CV makes efficient use of the available data for testing
- note that whenever we use multiple training sets, as in CV and random resampling, we are evaluating a learning method as opposed to an individual learned model

Confusion matrices / contingency tables

task: activity recognition from video



	bend	jack	jump	pjump	run	side	skip	walk	wave1	wave2
bend	100	0	0	0	0	0	0	0	0	0
jack	0	100	0	0	0	0	0	0	0	0
jump	0	0	89	0	0	0	11	0	0	0
pjump	0	0	0	100	0	0	0	0	0	0
run	0	0	0	0	89	0	11	0	0	0
side	0	0	0	0	0	100	0	0	0	0
skip	0	0	0	0	0	0	100	0	0	0
walk	0	0	0	0	0	0	0	100	0	0
wave1	0	0	0	0	0	0	0	0	67	33
wave2	0	0	0	0	0	0	0	0	0	100

predicted class

figure from vision.jhu.edu

Confusion matrix for 2-class problems

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

$$\text{error} = 1 - \text{accuracy} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Is accuracy perfect?

accuracy may not be useful measure in cases where

- there is a large class skew
  - Is 98% accuracy good when 97% of the instances are negative?
- there are differential misclassification costs – say, getting a positive wrong costs more than getting a negative wrong
  - Consider a medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease
- we are most interested in a subset of high-confidence predictions

Other performance metrics

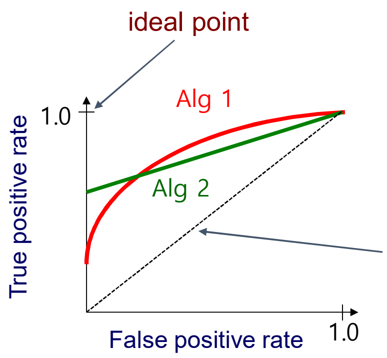
		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{true positive rate (recall)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{false positive rate} = \frac{\text{FP}}{\text{actual neg}} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

ROC curve

A *Receiver Operating Characteristic* (*ROC*) curve plots the TP-rate vs. the FP-rate as a threshold on the confidence of an instance being positive is varied

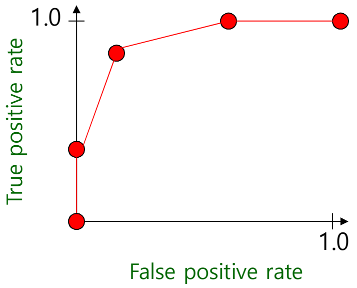


expected curve for random guessing

Different methods can work better in different parts of ROC space.

Plotting an ROC curve

instance	confidence positive		correct class
Ex 9	.99		+
Ex 7	.98	TPR= 2/5, FPR= 0/5	+
Ex 1	.72		-
Ex 2	.70		+
Ex 6	.65	TPR= 4/5, FPR= 1/5	+
Ex 10	.51		-
Ex 3	.39		-
Ex 5	.24	TPR= 5/5, FPR= 3/5	+
Ex 4	.11		-
Ex 8	.01	TPR= 5/5, FPR= 5/5	-



ROC curve example

task: recognizing genomic units called operons

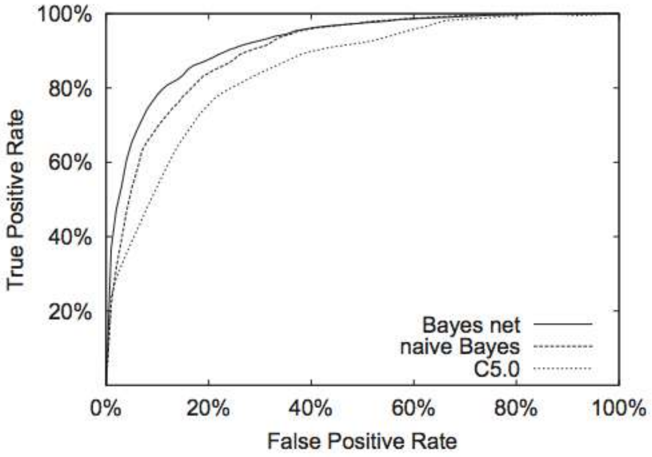
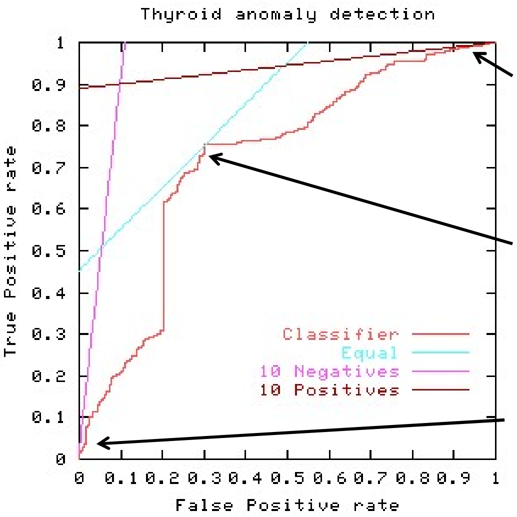


figure from Bockhorst et al., *Bioinformatics* 2003

Plotting an ROC example

The best operating point depends on the relative costs of FN and FP misclassifications



best operating point when
FN costs 10× FP

best operating point when
cost of misclassifying positives and n
egatives is equal

best operating point when
FP costs 10× FN

ROC curve

Does a low false-positive rate indicate that most positive predictions (i.e. predictions with confidence > some threshold) are correct?

suppose our TPR is 0.9, and FPR is 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

Other performance metrics

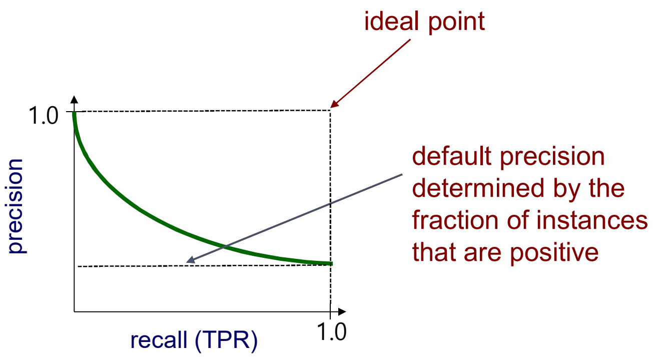
		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{recall (TP rate)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{precision (positive predictive value)} = \frac{\text{TP}}{\text{predicted pos}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision / recall curve

A *precision/recall curve* plots the precision vs. recall (TP-rate) as a threshold on the confidence of an instance being positive is varied



ideal point

default precision determined by the fraction of instances that are positive

Precision / recall curve example

predicting patient risk for VTE

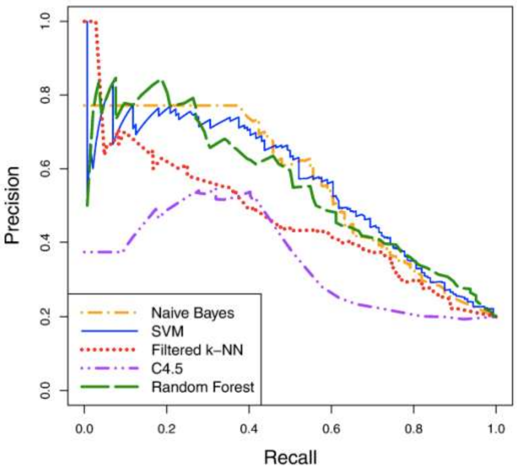


figure from Kawaler et al., *Proc. of AMIA Annual Symosium,* 2012

How do we get one ROC/PR curve when we do CV?

Approach 1

- make assumption that confidence values are comparable across folds
- pool predictions from all test sets
- plot the curve from the pooled predictions

Approach 2 (for ROC curves)

- plot individual curves for all test sets
- view each curve as a function
- plot the average curve for this set of functions

Comments…

Both

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

ROC curves

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

Precision/recall curves

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