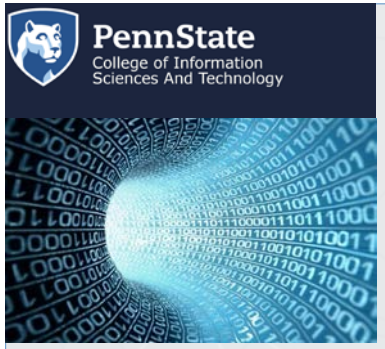


PennState College of Information Sciences And Technology

Center for Big Data Analytics and Discovery Informatics
Artificial Intelligence Research Laboratory

CTSI Clinical and Translational Science Institute

Evaluating Classifier Performance

Vasant Honavar
Artificial Intelligence Research Laboratory
Informatics Graduate Program
Computer Science and Engineering Graduate Program
Bioinformatics and Genomics Graduate Program
Neuroscience Graduate Program
Center for Big Data Analytics and Discovery Informatics
Huck Institutes of the Life Sciences
Institute for Cyberscience
Clinical and Translational Sciences Institute
Northeast Big Data Hub
Pennsylvania State University

vhonavar@ist.psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu

Why Evaluate classifiers?

- To know how well a classifier can be expected to perform when it is put to use
- To choose the best model from among a set of alternatives

Evaluating a Classifier

- How can we measure performance of classifiers?
- How well can a classifier be expected to perform on *novel* data, i.e., data not seen during training?
- We can *estimate* the *performance* (e.g., accuracy, sensitivity) of the classifier using an evaluation data set (not used for training)
- How close is the *estimated* performance to the *true* performance?

Classification error

- Error = classifying a record as belonging to one class when it belongs to another class.
- Error rate = percent of misclassified samples out of the total samples in the validation data

Naïve Baseline

Naïve baseline: classify all samples as belonging to the most prevalent class

- We hope to do better than the naïve baseline
- When the goal is to identify high-value but rare outcomes, we may do well by doing worse than the naïve baseline in terms of accuracy

Estimating Classifier Performance

N: Total number of instances in the data set

TP_j: Number of True positives for class j

FP_j : Number of False positives for class j

TN_j: Number of True Negatives for class j

FN_j: Number of False Negatives for class j

$$Accuracy_j = \frac{TP_j + TN_j}{N}$$
$$= P(class = c_j \wedge label = c_j)$$

Perfect classifier ←→ Accuracy =1

Popular measure

Biased in favor of the majority class!

Should be used with caution!

Classifier Learning -- Measuring Performance

Class Label →	C_1	$\neg C_1$
C_1 ↓	TP= 55	FP=5
$\neg C_1$	FN=10	TN=30

$$N = TP + FN + TN + FP = 100$$

$$sensitivity_1 = \frac{TP}{TP + FN} = \frac{55}{55 + 10} = \frac{55}{65}$$

$$specificity_1 = \frac{TP}{TP + FP} = \frac{55}{55 + 5} = \frac{55}{60}$$

$$accuracy_1 = \frac{TP + TN}{N} = \frac{55 + 30}{100} = \frac{85}{100}$$

$$falsealarm_1 = \frac{FP}{TN + FP} = \frac{5}{30 + 5} = \frac{5}{35}$$

When One Class is More Important than another

In many cases it is more important to identify members of a specific target class

- Tax fraud
- Credit default
- Response to promotional offer
- Detecting electronic network intrusion
- Predicting delayed flights
- Diagnosing cancer
- Predicting nuclear reactor meltdown

In such cases, we may tolerate greater overall error, in return for better predictions of the more important class

Measuring Classifier Performance: Sensitivity

$$Sensitivity_j = \frac{TP_j}{TP_j + FN_j}$$
$$= \frac{Count(label = c_j \wedge class = c_j)}{Count(class = c_j)}$$
$$= P(label = c_j \mid class = c_j)$$

Perfect classifier → Sensitivity = 1

Probability of correctly labeling members of the target class

Also called recall or hit rate

Classifier Learning -- Measuring Performance

Class Label →	C_1	$\neg C_1$
C_1 ↓	TP= 55	FP=5
$\neg C_1$	FN=10	TN=30

$$N = TP + FN + TN + FP = 100$$

$$sensitivity_1 = \frac{TP}{TP + FN} = \frac{55}{55 + 10} = \frac{55}{65}$$

$$specificity_1 = \frac{TP}{TP + FP} = \frac{55}{55 + 5} = \frac{55}{60}$$

$$accuracy_1 = \frac{TP + TN}{N} = \frac{55 + 30}{100} = \frac{85}{100}$$

$$falsealarm_1 = \frac{FP}{TN + FP} = \frac{5}{30 + 5} = \frac{5}{35}$$

Measuring Classifier Performance: Specificity

$$Specificity_j = \frac{TP_j}{TP_j + FP_j}$$
$$= \frac{Count(label = c_j \wedge class = c_j)}{Count(label = c_j)}$$
$$= P(class = c_j \mid label = c_j)$$

Perfect classifier → Specificity = 1

Also called precision

Probability that a positive prediction is correct

Measuring Performance: Precision, Recall, and False Alarm Rate

$$Precision_j = Specificity_j = \frac{TP_j}{TP_j + FP_j}$$

$$Recall_j = Sensitivity_j = \frac{TP_j}{TP_j + FN_j}$$

Perfect classifier → Precision=1 Perfect classifier → Recall=1

$$FalseAlarm_j = \frac{FP_j}{TN_j + FP_j}$$
$$= \frac{Count(label = c_j \wedge class = \neg c_j)}{Count(label = \neg c_j)}$$
$$= P(label = c_j \mid class = \neg c_j)$$

Perfect classifier → False Alarm Rate = 0

Classifier Learning -- Measuring Performance

Class Label →	C_1	$\neg C_1$
C_1 ↓	TP= 55	FP=5
$\neg C_1$	FN=10	TN=30

$$N = TP + FN + TN + FP = 100$$

$$sensitivity_1 = \frac{TP}{TP + FN} = \frac{55}{55 + 10} = \frac{55}{65}$$

$$specificity_1 = \frac{TP}{TP + FP} = \frac{55}{55 + 5} = \frac{55}{60}$$

$$accuracy_1 = \frac{TP + TN}{N} = \frac{55 + 30}{100} = \frac{85}{100}$$

$$falsealarm_1 = \frac{FP}{TN + FP} = \frac{5}{30 + 5} = \frac{5}{35}$$

Measuring Performance – Correlation Coefficient

$$CC_j = \frac{(TP_j \times TN_j) - (FP_j \times FN_j)}{\sqrt{(TP_J + FN_j)(TP_j + FP_j)(TN_j + FP_j)(TN_j + FN_j)}}$$

$$-1 \le CC_j \le 1$$

$$CC_j = \sum_{d_i \in D} \frac{(jlabel_i - \overline{jlabel})(jclass_i - \overline{jclass})}{\sigma_{JLABEL}\sigma_{JCLASS}}$$

where $jlabel_i = 1$ iff the classifier assigns d_i to class c_j

$jclass_i = 1$ iff the true class of d_i is class c_j

Beware of terminological confusion in the literature!

- Some bioinformatics authors use "accuracy" incorrectly to refer to recall i.e. sensitivity or precision i.e. specificity
- In medical statistics, specificity sometimes refers to sensitivity for the negative class i.e.

$$\frac{TN_j}{TN_j + FP_j}$$

- Some authors use false alarm rate to refer to the probability that a positive prediction is incorrect i.e.

$$\frac{FP_j}{FP_j + TP_j} = 1 - Precision_j$$

When you write

- provide the formula in terms of TP, TN, FP, FN

When you read

- check the formula in terms of TP, TN, FP, FN

Measuring Classifier Performance

- TP, FP, TN, FN provide the relevant information
- No single measure tells the whole story
- A classifier with 98% accuracy can be useless if 98% of the population does not have cancer and the 2% that do are misclassified by the classifier
- Use of multiple measures recommended
- Beware of terminological confusion!

Micro-averaged performance measures
Performance on a random sample

- Micro averaging gives equal importance to each sample
- Classes with large number of instances dominate

$$MicroAverage\ Precision = \frac{\sum_j TP_j}{\sum_j TP_j + \sum_j FP_j}$$

$$MicroAverage\ Recall = \frac{\sum_j TP_j}{\sum_j TP_j + \sum_j FN_j}$$

$$MicroAverage\ FalseAlarm = 1 - MicroAverage\ Precision$$

$$MicroAverage\ Accuracy = \frac{\sum_j TP_j}{N}$$

Etc.

$$MicroAverage\ CC = \frac{\left(\left(\sum_j TP_j\right) \times \left(\sum_j TN_j\right)\right) - \left(\left(\sum_j FP_j\right) \times \left(\sum_j FN_j\right)\right)}{\sqrt{\left(\sum_j TP_j + \sum_j FN_j\right)\left(\sum_j TP_j + \sum_j FP_j\right)\left(\sum_j TN_j + \sum_j FP_j\right)\left(\sum_j TN_j + \sum_j FN_j\right)}}$$

Macro-averaged performance measures

Macro averaging gives equal importance to each of the M classes

$$MacroAverage\ Sensitivity = \frac{1}{M}\sum_j Sensitivity_j$$

$$MacroAverageCorrelationCoeff = \frac{1}{M}\sum_j CorrelationCoeff_j$$

$$MacroAverage\ Specificity = \frac{1}{M}\sum_j Specificity_j$$

Cutoff for classification

Most machine learning algorithms classify via a 2-step process:

For each sample,

1. Compute **probability of belonging to class "1"**
2. Compare to cutoff value, and classify accordingly

- Default cutoff value is 0.50

  If >= 0.50, classify as "1"

  If < 0.50, classify as "0"

- Can use different cutoff values for trading off one measure against another (more on this later)

- Question: How would this work in the case of K nearest neighbor?

Cutoff Table

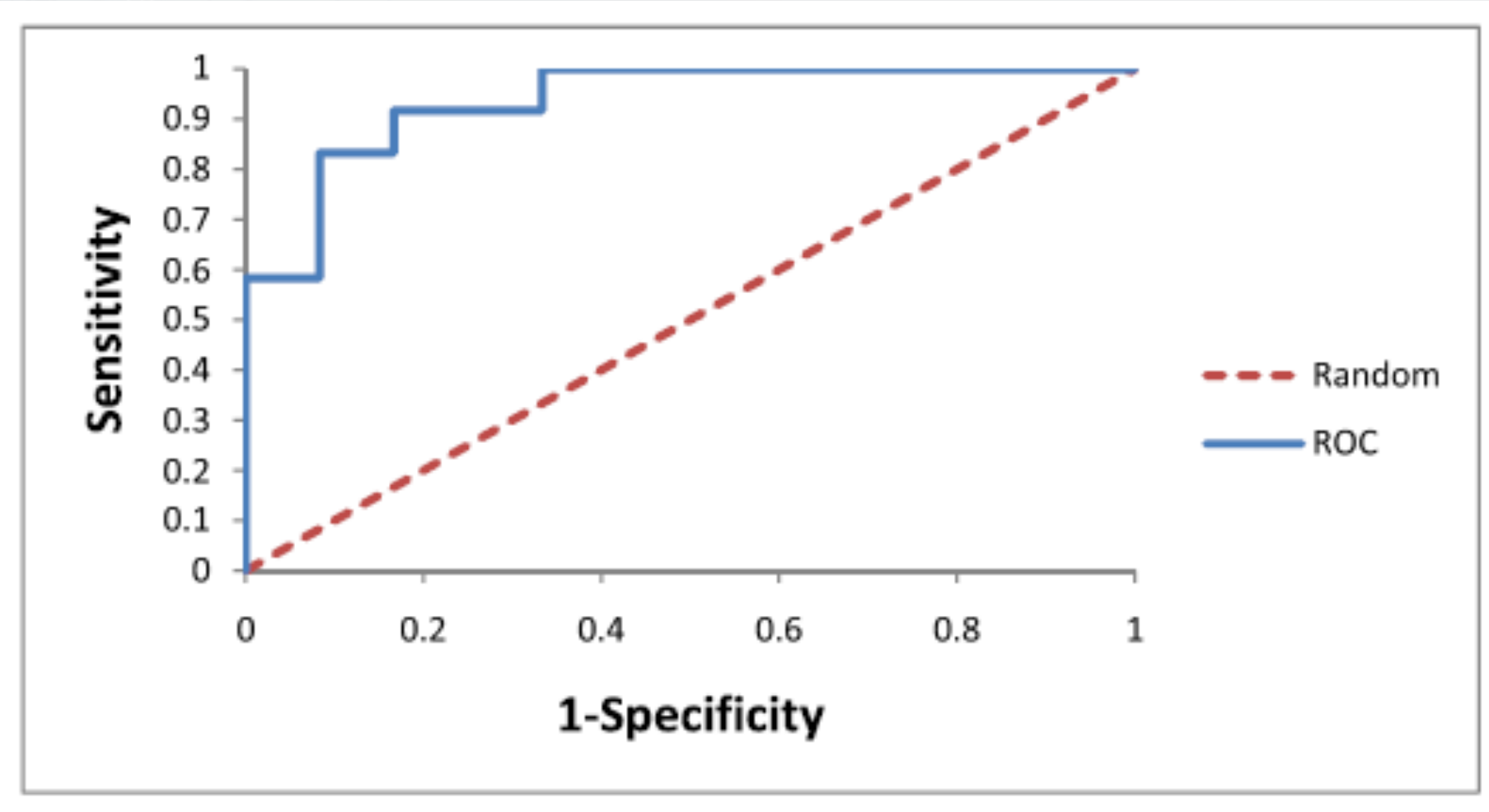
Actual Class	Prob. of "1"	Actual Class	Prob. of "1"
1	0.996	1	0.506
1	0.988	0	0.471
1	0.984	0	0.337
1	0.980	1	0.218
1	0.948	0	0.199
1	0.889	0	0.149
1	0.848	0	0.048
0	0.762	0	0.038
1	0.707	0	0.025
1	0.681	0	0.022
1	0.656	0	0.016
0	0.622	0	0.004

- If cutoff is 0.50: 12 samples are classified as “1”
- If cutoff is 0.80: seven samples are classified as “1”

Receiver Operating Characteristic (ROC) Curve

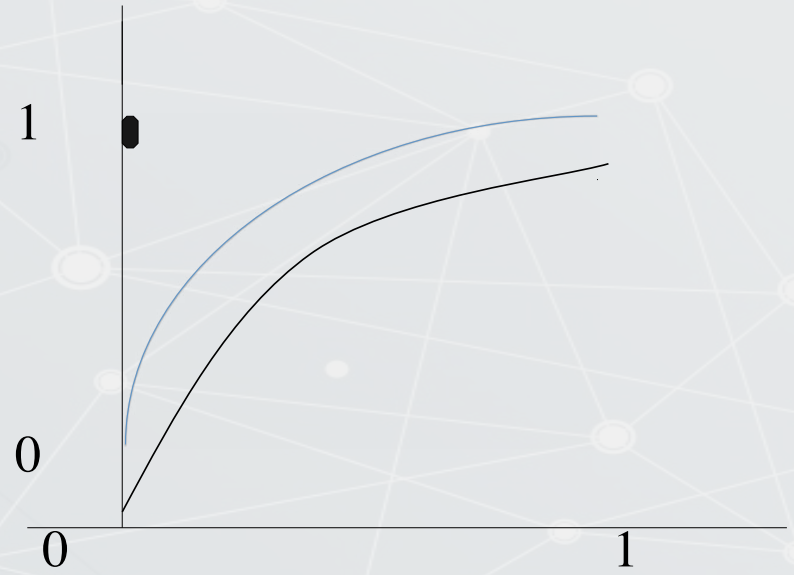
- The confusion matrix, and hence the previous measures of classifier performance are threshold dependent
- We can often trade off recall versus precision – e.g., by adjusting classification threshold θ
- Is there a threshold-independent measure of classifier performance?
  - ROC curve is a plot of Sensitivity against False Alarm Rate which is same as (1-Specificity) which characterizes this tradeoff for a given classifier
  - ROC curve is obtained by plotting sensitivity against (1-specificity) by varying the classification threshold

Receiver operating characteristic (ROC) Curve

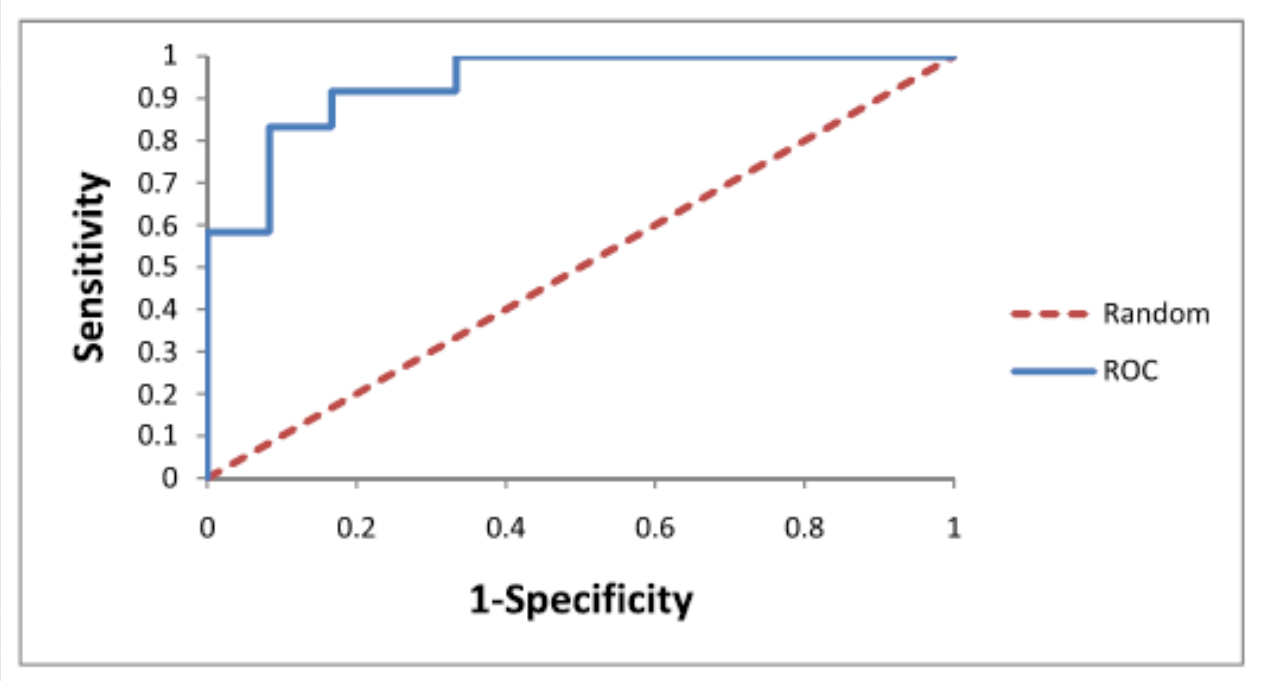


Measuring Performance of Classifiers – ROC curves

- ROC curves offer a more complete picture of the performance of the classifier as a function of the classification threshold
- A classifier *h* is better than another classifier *g* if ROC(*h*) dominates the ROC(*g*)
- ROC(*h*) dominates ROC(*g*) → AreaROC(*h*) > AreaROC(*g*)



ROC Curve


Sensitivity
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1-Specificity
Random
ROC

Misclassification Costs May Differ

- The cost of making a misclassification error may be higher for one class than the other(s)
- Looked at another way, the benefit of making a correct classification may be higher for one class than the other(s)

Example – Response to Promotional Offer

- Suppose we send an offer to 1000 people, with 1% average response rate
- "1" = response, "0" = nonresponse

- "Naïve rule" (classify everyone as "0") has error rate of 1% (seems good)
- Using machine learning suppose we can correctly classify eight 1's as 1's
- But at the cost of misclassifying twenty 0's as 1's and two 1's as 0's.

Confusion Matrix

	Predict as 1	Predict as 0
Actual 1	8	2
Actual 0	20	970

Error rate = (2+20) = 2.2% (higher than naïve rate)

Introducing Costs & Benefits

Suppose:

- Profit from a "1" is $10
- Cost of sending offer is $1

Then:

- Under naïve rule, all are classified as "0", so no offers are sent: no cost, no profit
- Under DM predictions, 28 offers are sent.

  8 respond with profit of $10 each

  20 fail to respond, cost $1 each

  972 receive nothing (no cost, no profit)

Profit Matrix

	Predict as 1	Predict as 0
Actual 1	$80	0
Actual 0	($20)	0

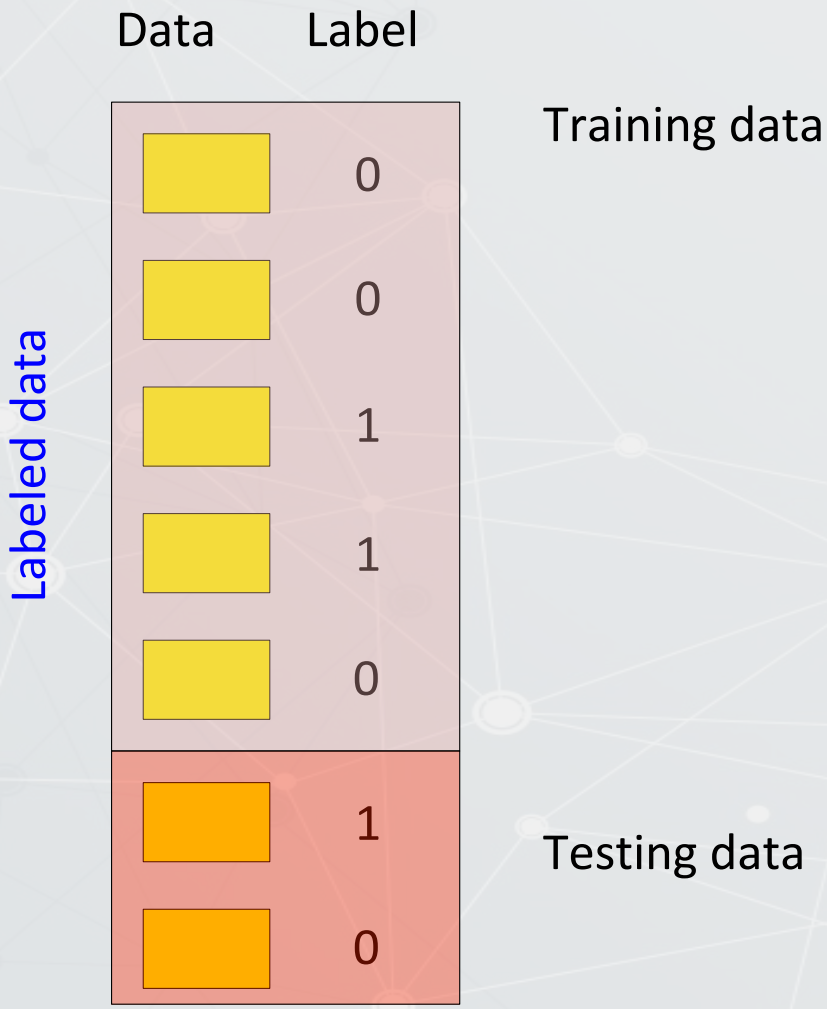
Evaluating a Classifier

- What we have done so far is to estimate the classifier's performance on some available data.
- How well can a classifier be expected to perform on *novel* data?
- Performance estimated on training data is often optimistic relative to performance on novel data
- We can *estimate* the *performance* (e.g., accuracy, sensitivity) of the classifier using evaluation data (not used for training)
- How close is the *estimated* performance to the *true* performance?

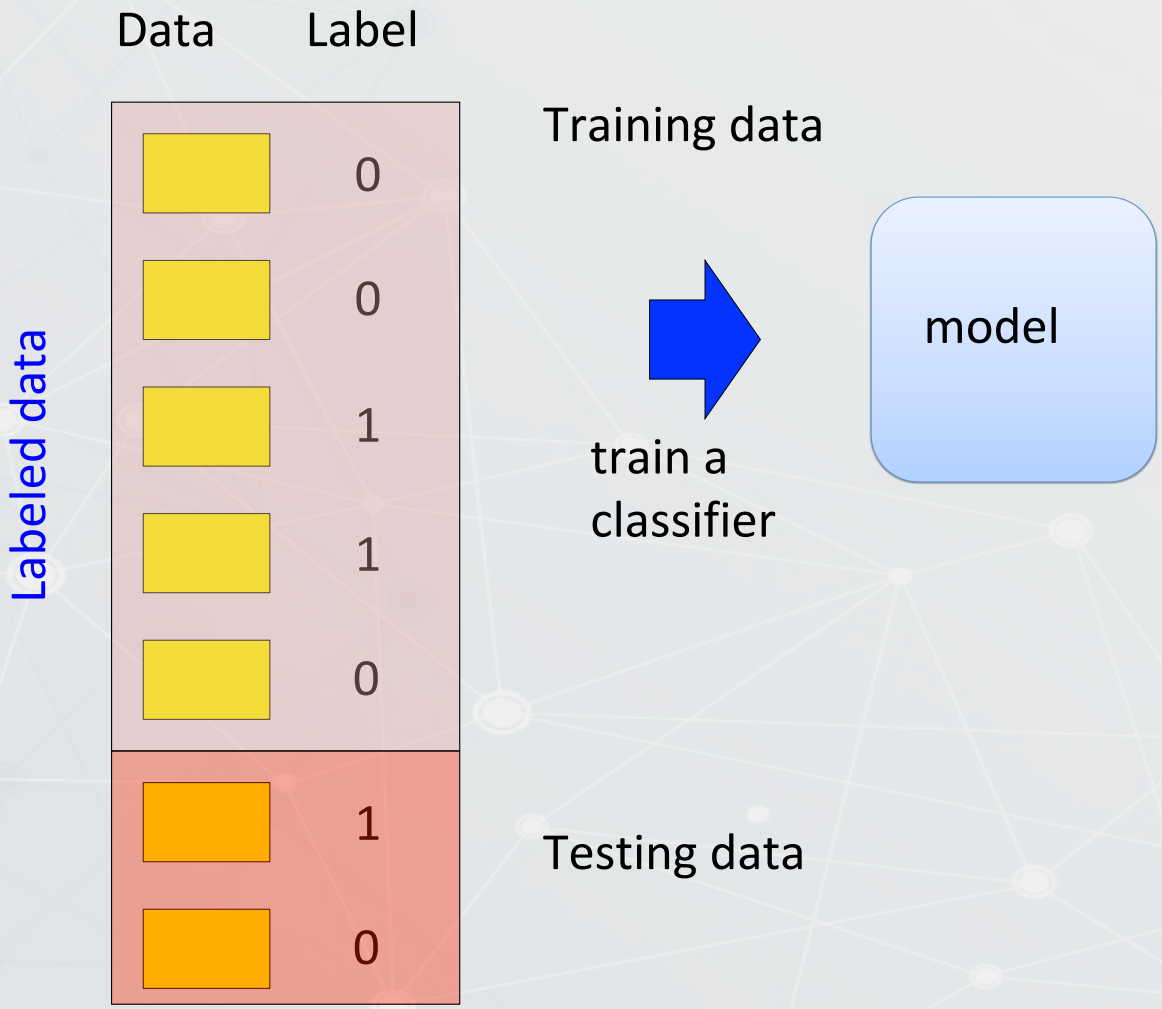
Evaluation of a classifier with limited data

- Holdout method – use part of the data for training, and the rest for testing
- We may be lucky or unlucky – training data or test data may not be *representative*
- Solution – Run multiple experiments with disjoint training and test data sets in which each class is represented in roughly the same proportion as in the entire data set

Classifier evaluation

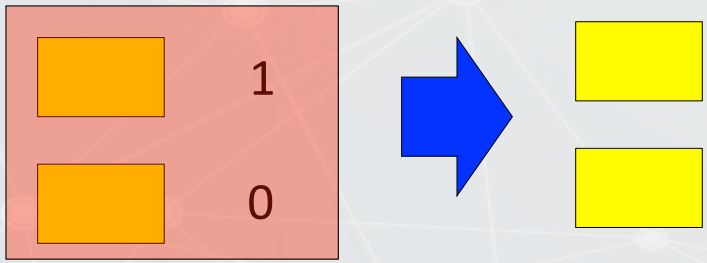


Classifier evaluation



Classifier evaluation

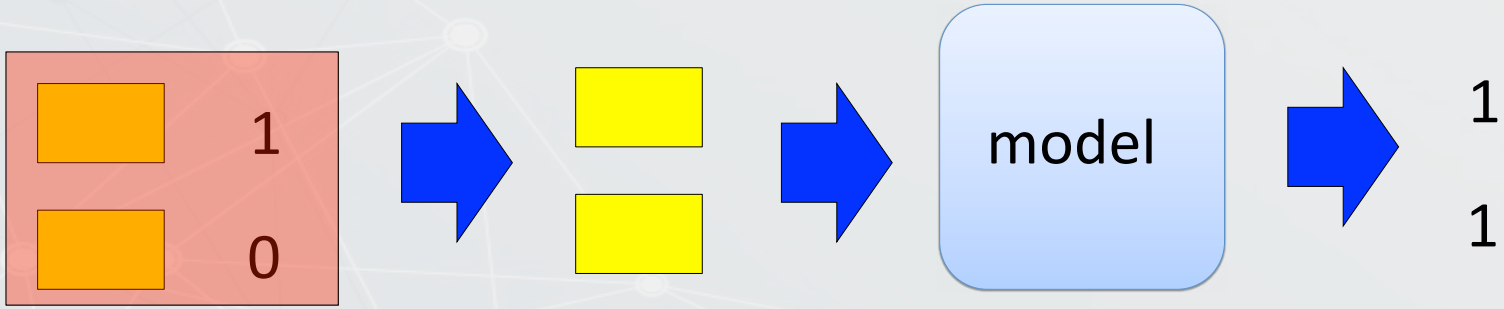
Data Label



Pretend like we don't know the labels

Classifier evaluation

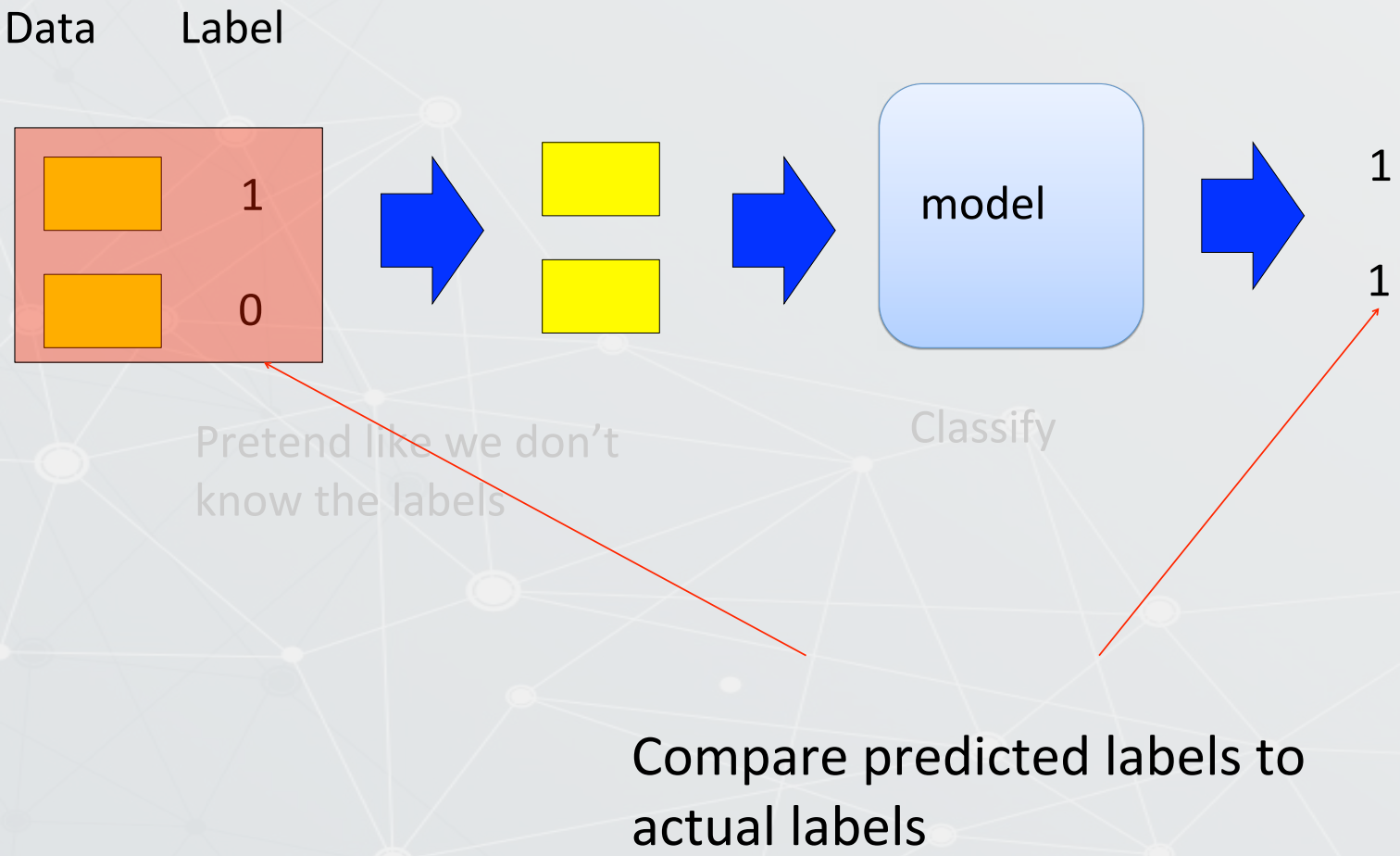
Data Label



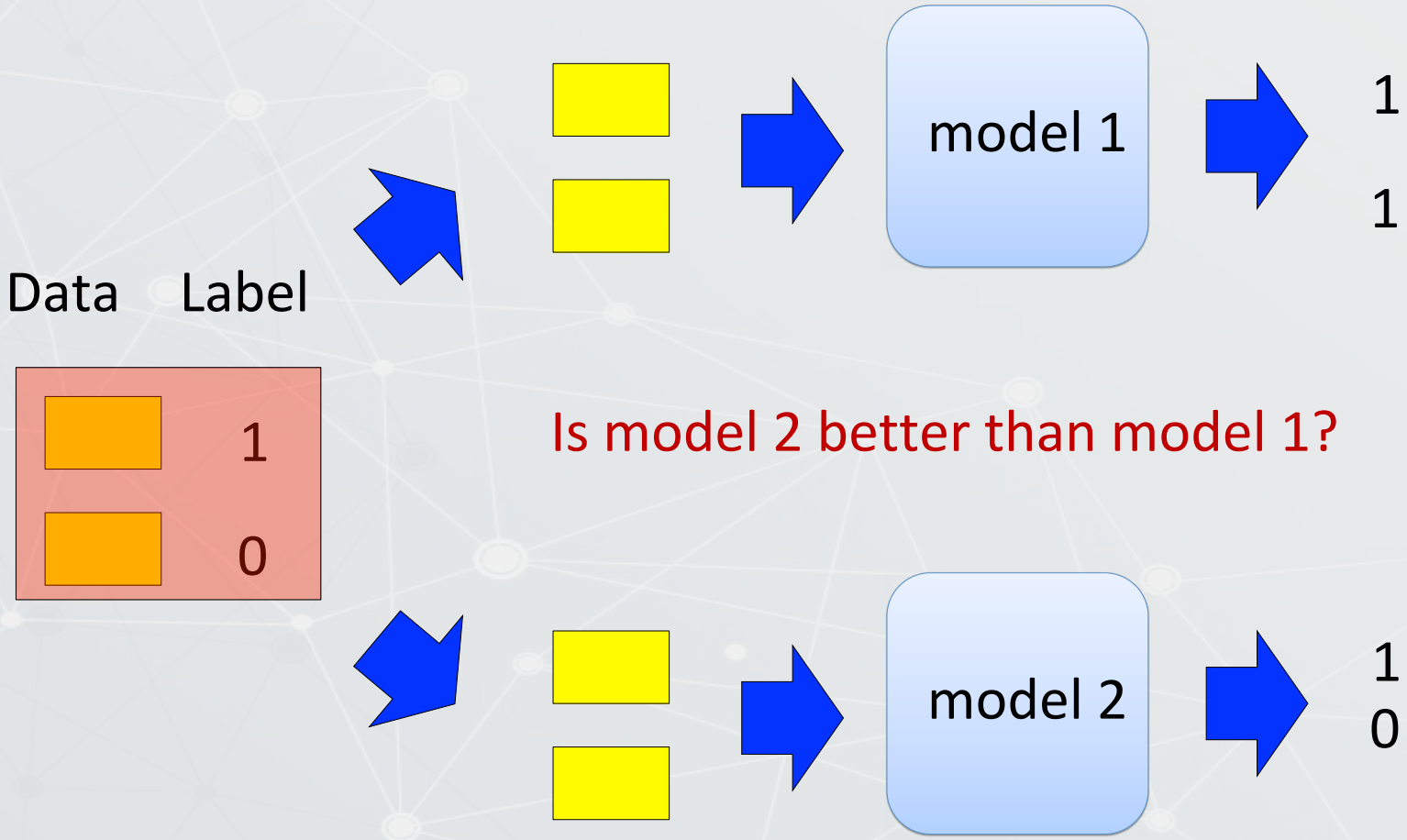
Classify

Pretend like we don't know the labels

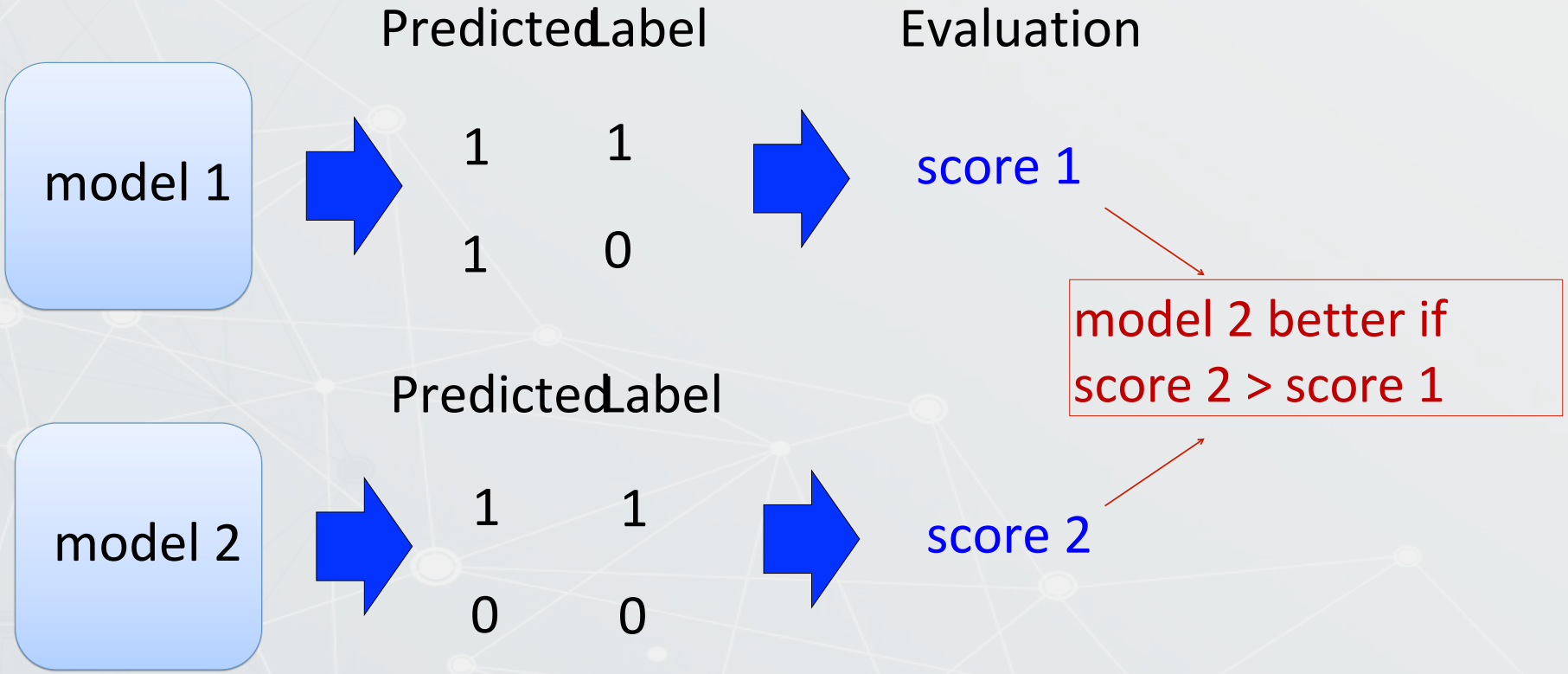
Classifier evaluation



Comparing algorithms



Comparing algorithms



When would we want to do this type of comparison?

Is model 2 better?

Model 1: 85% accuracy
Model 2: 80% accuracy

Model 1: 85.5% accuracy
Model 2: 85.0% accuracy

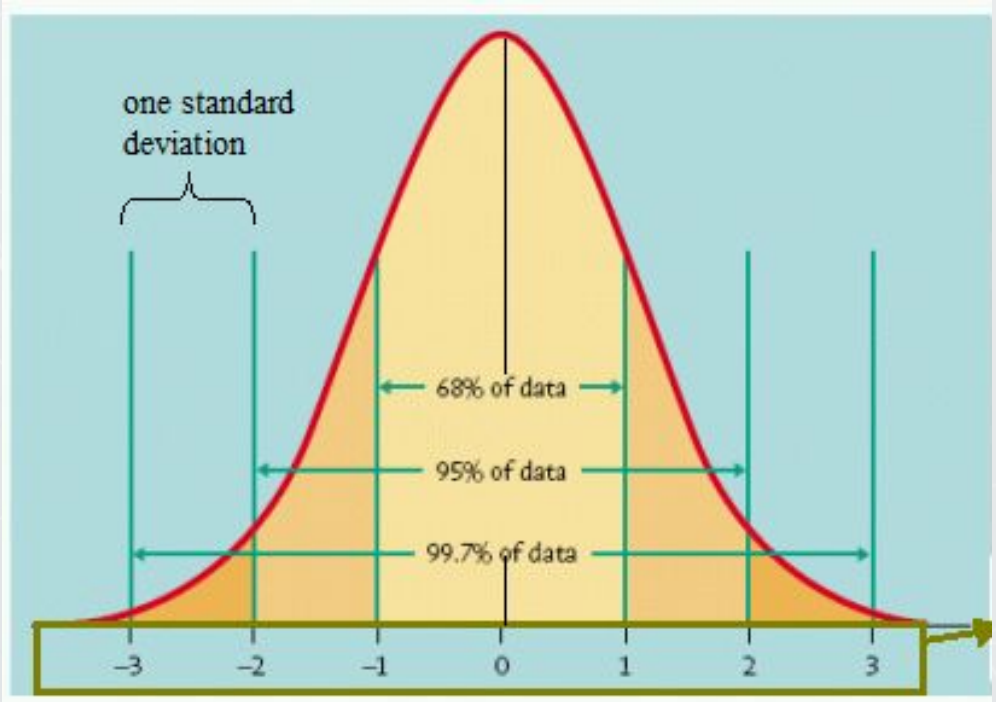
Model 1: 0% accuracy
Model 2: 100% accuracy

Comparing scores: significance

- Just comparing scores on one data set isn't enough!
- We don't just want to know which system is better on one particular data set, we want to know if model 1 is better than model 2 in general
- Put another way, we want to be confident that the difference is real and not just due to random chance

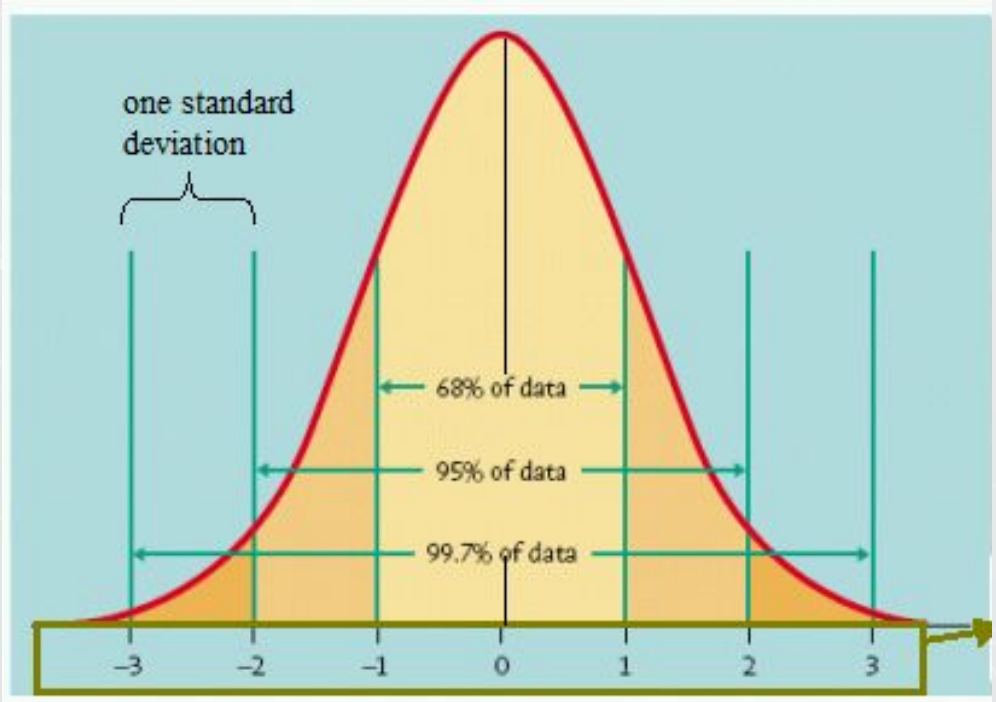
How do we know how variable a model's accuracy is?

Variance

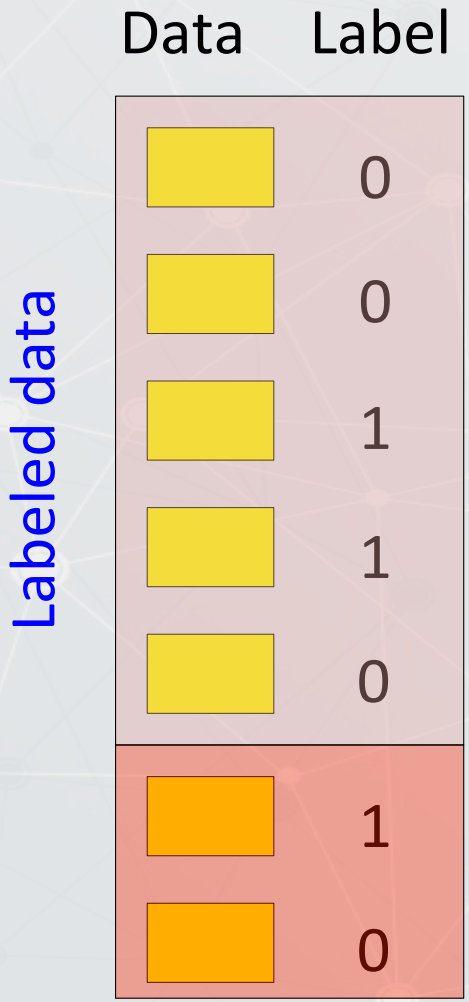


Variance of performance

- We need multiple accuracy scores!
- How can we get them?

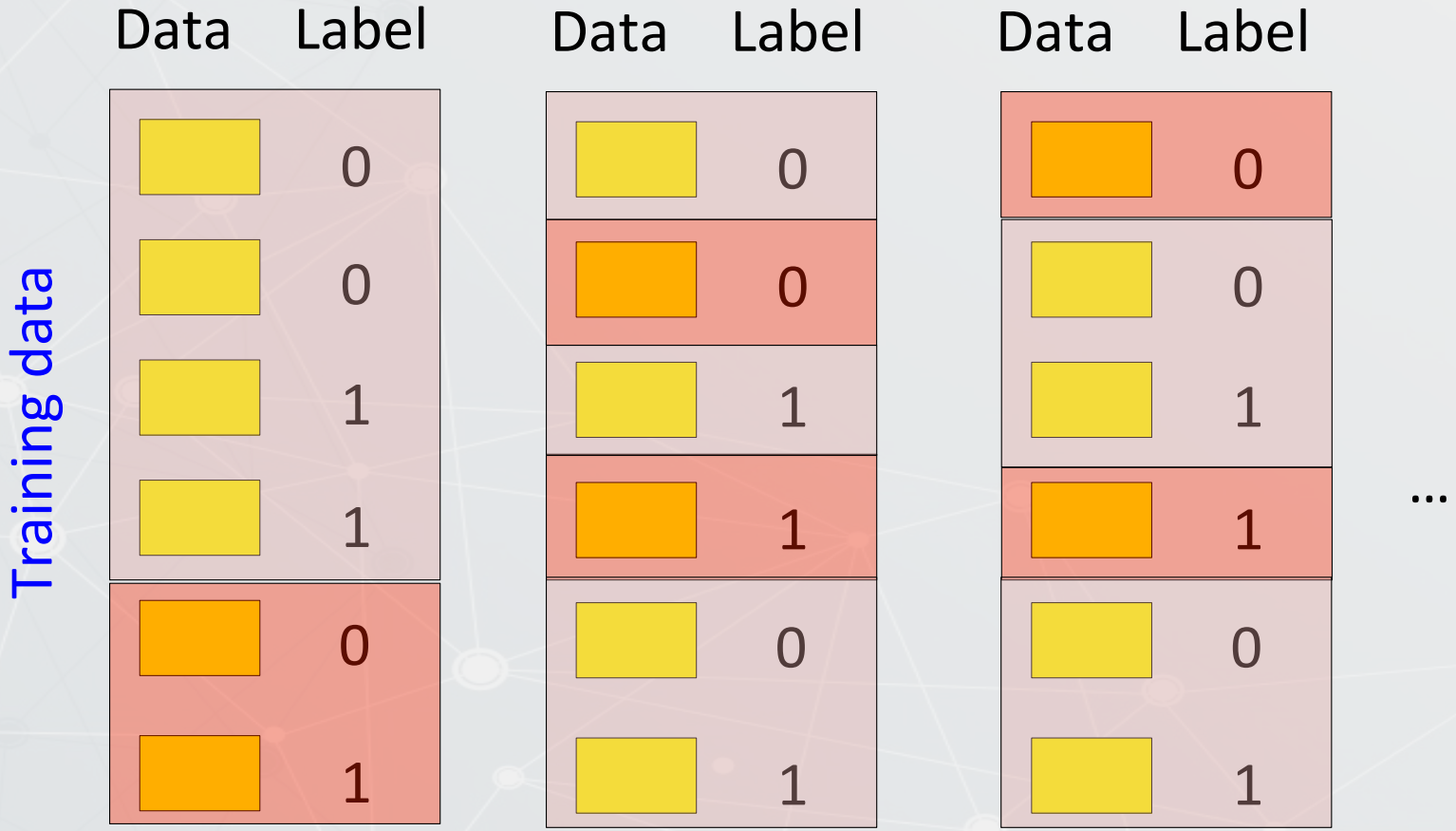


Repeated experimentation



Instead of one evaluation with a particular split of training and test data, run multiple evaluations, with different splits of training and test data

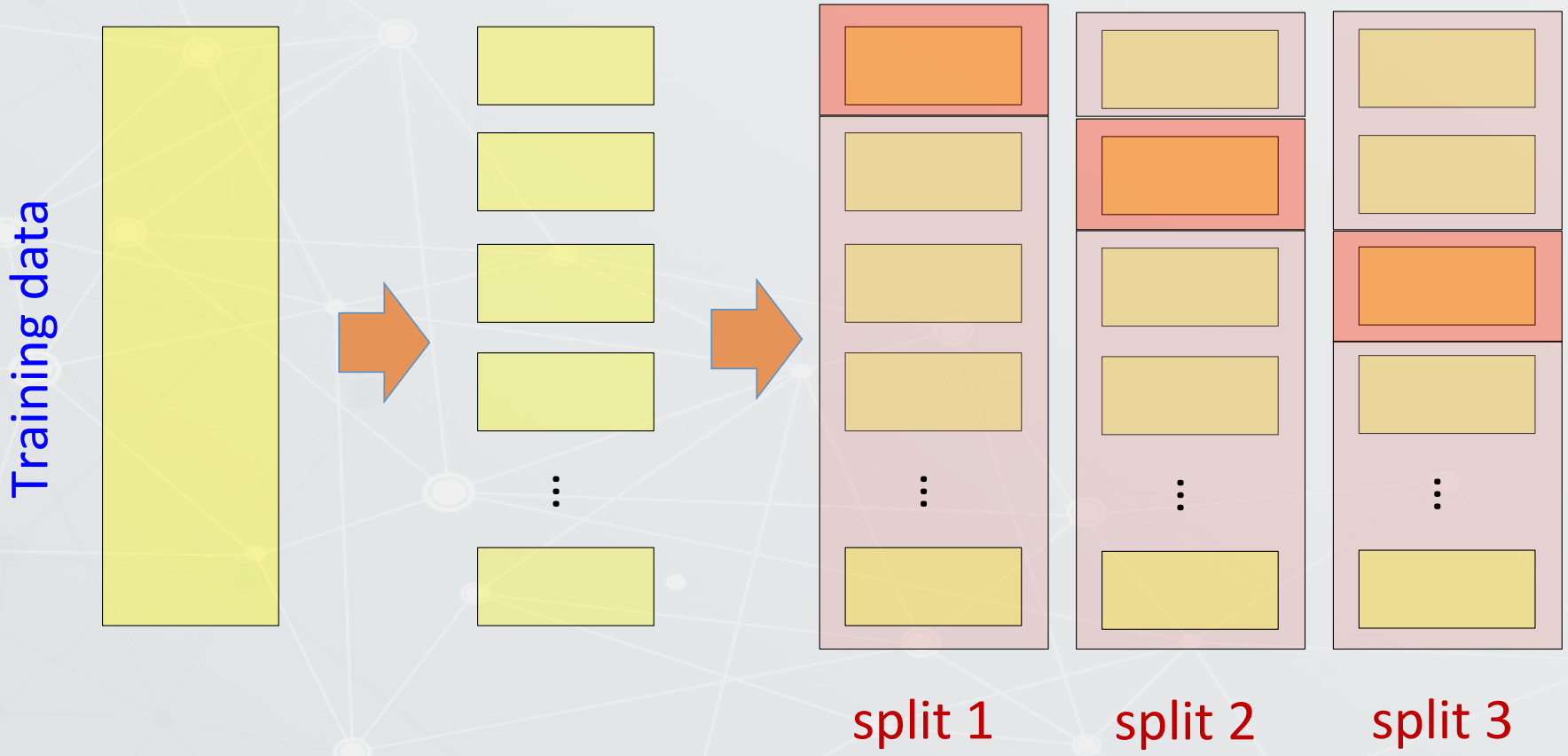
Repeated experimentation



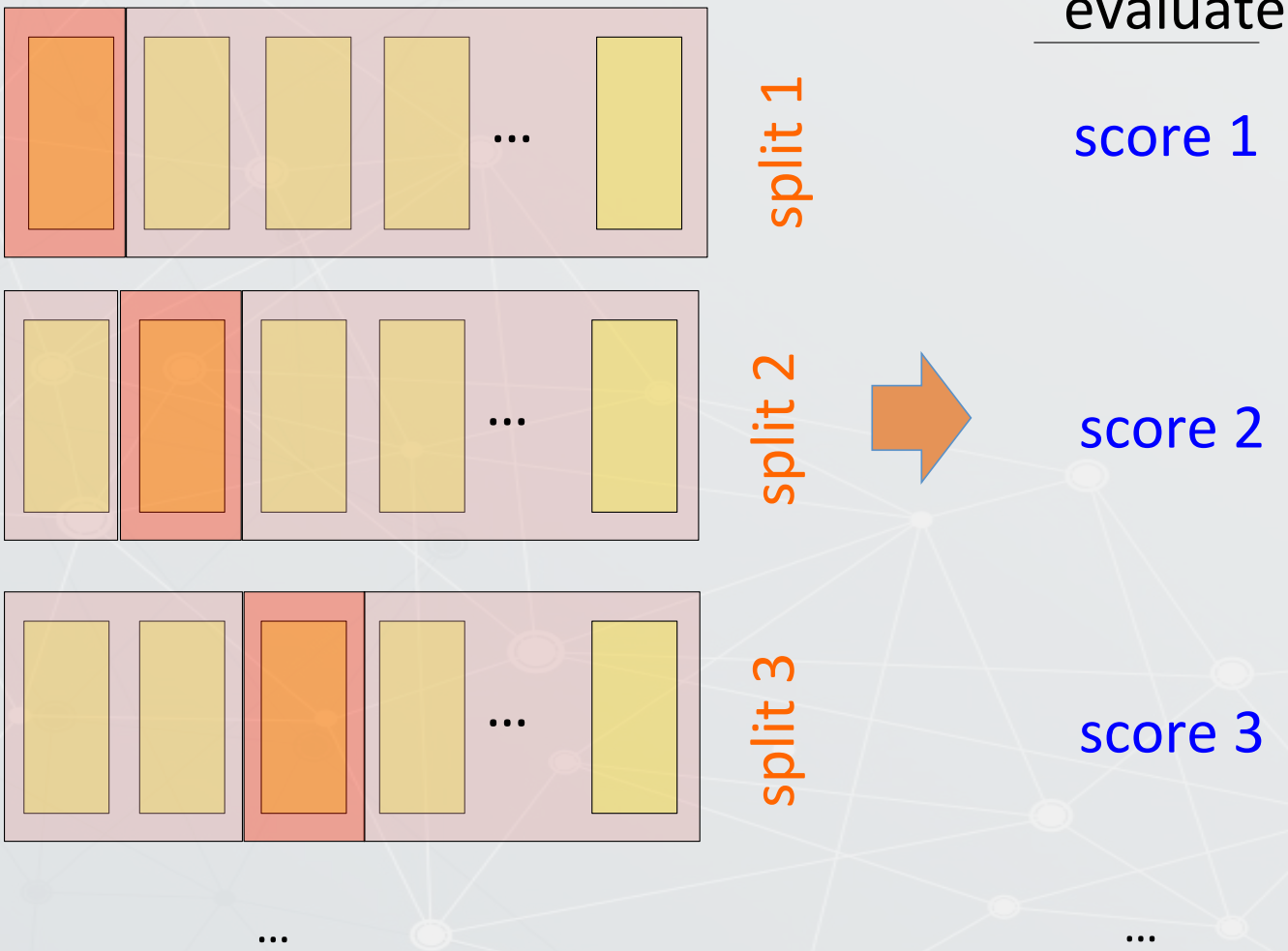
K-fold cross validation

break into n equal-sized parts

repeat for all parts/splits:
train on K-1 parts evaluate on the other



K-fold cross validation



K-fold cross validation

- Better utilization of labeled data
- More robust: don't just rely on one evaluation set to evaluate the approach (or for optimizing parameters)
- Multiplies the computational overhead by *K* (have to train *K* models instead of just one)
- 10 is the most common choice of *K*

Estimating the performance of a classifier

K-fold cross-validation

Partition the data (multi) set *S* into *K* equal parts $S_1 .. S_K$

with roughly the same class distribution as *S*.

Errorc = 0

For *i*=1 to *K* do

$$\{ \quad S_{Test} \leftarrow S_i \;\; S_{Train} \leftarrow S - S_i;$$
$$\alpha \leftarrow Learn(S_{Train})$$
$$Errorc \leftarrow Errorc + Error(\alpha, S_{Test}) \}$$

$$Error \leftarrow \left(\frac{Errorc}{K} \right); \quad Output(Error)$$

Estimating classifier performance

Recommended procedure

- Use *K*-fold cross-validation (*K*=5 or 10) for estimating performance estimates (accuracy, precision, recall, points on ROC curve, etc.) and 95% confidence intervals around the mean
- Compute mean values of performance estimates and standard deviations of performance estimates
- Report mean values of performance estimates and their standard deviations or 95% confidence intervals around the mean
- Be skeptical – repeat experiments several times with different random splits of data into *K* folds!

Leave-one-out cross validation

- *K*-fold cross validation where *K* = number of samples
- aka “jackknifing”
- pros/cons?
- when would we use this?

Leave-one-out cross-validation

- *K*-fold cross validation with *K* = *n* where *n* is the total number of samples available
- *n* experiments – using *n*-1 samples for training and the remaining sample for testing
- Leave-one-out cross-validation does not guarantee the same class distribution in training and test data!

  Extreme case: 50% class 1, 50% class 2

  Predict majority class label in the training data

  True error – 50%;

  Leave-one-out error estimate – 100%!!!!!

Leave-one-out cross validation

- Can be very expensive if training is slow and/or if there are a large number of examples
- Useful in domains with limited training data: *maximizes the data we can use for training*
- Some classifiers permit the estimation of leave-1-out performance measure without actually having to train K models

Comparing systems: sample 1

split	model 1	model 2
1	87	88
2	85	84
3	83	84
4	80	79
5	88	89
6	85	85
7	83	81
8	87	86
9	88	89
10	84	85
average:	**85**	**85**

Is model 2 better than model 1?

Comparing systems: sample 2

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
avg	**82**	**85**

Is model 2 better than model 1?

Comparing systems: sample 3

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**

Is model 2 better than model 1?

Comparing systems

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
average:	**82**	**85**

What's the difference?

Comparing systems

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**
std dev	**2.3**	**1.7**

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
average:	**82**	**85**
std dev	**5.9**	**3.9**

Comparing systems: sample 4

Is model 2 better than model 1?

split	model 1	model 2
1	80	82
2	84	87
3	89	90
4	78	82
5	90	91
6	81	83
7	80	80
8	88	89
9	76	77
10	86	88
average	**83**	**85**
std dev	**4.9**	**4.7**

Comparing systems: sample 4

split	model 1	model 2	model 2 – model 1
1	80	82	2
2	84	87	3
3	89	90	1
4	78	82	4
5	90	91	1
6	81	83	2
7	80	80	0
8	88	89	1
9	76	77	1
10	86	88	2
average	**83**	**85**	
std dev	**4.9**	**4.7**	

Is model 2 better than model 1?

split	model 1	model 2	model 2 – model 1
1	80	82	2
2	84	87	3
3	89	90	1
4	78	82	4
5	90	91	1
6	81	83	2
7	80	80	0
8	88	89	1
9	76	77	1
10	86	88	2
average	**83**	**85**	
std dev	**4.9**	**4.7**	

Comparing systems: sample 4

Model 2 is ALWAYS better

Comparing systems: sample 4

split	model 1	model 2	model 2 – model 1
1	80	82	2
2	84	87	3
3	89	90	1
4	78	82	4
5	90	91	1
6	81	83	2
7	80	80	0
8	88	89	1
9	76	77	1
10	86	88	2
average:	**83**	**85**	
std dev	**4.9**	**4.7**	

How do we decide if model 2 is better than model 1?

Statistical tests

Setup:

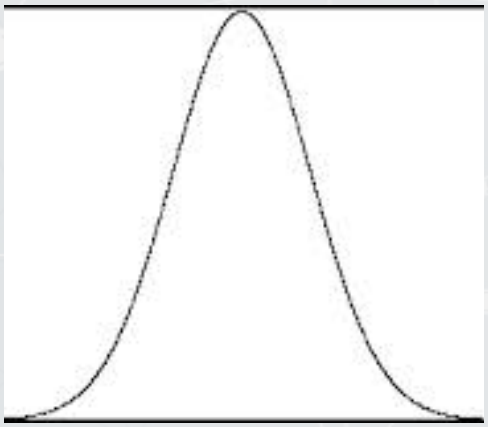
- Assume some default hypothesis about the data that you'd like to *disprove*, called the null hypothesis
- e.g. model 1 and model 2 are not statistically different in performance

Test:

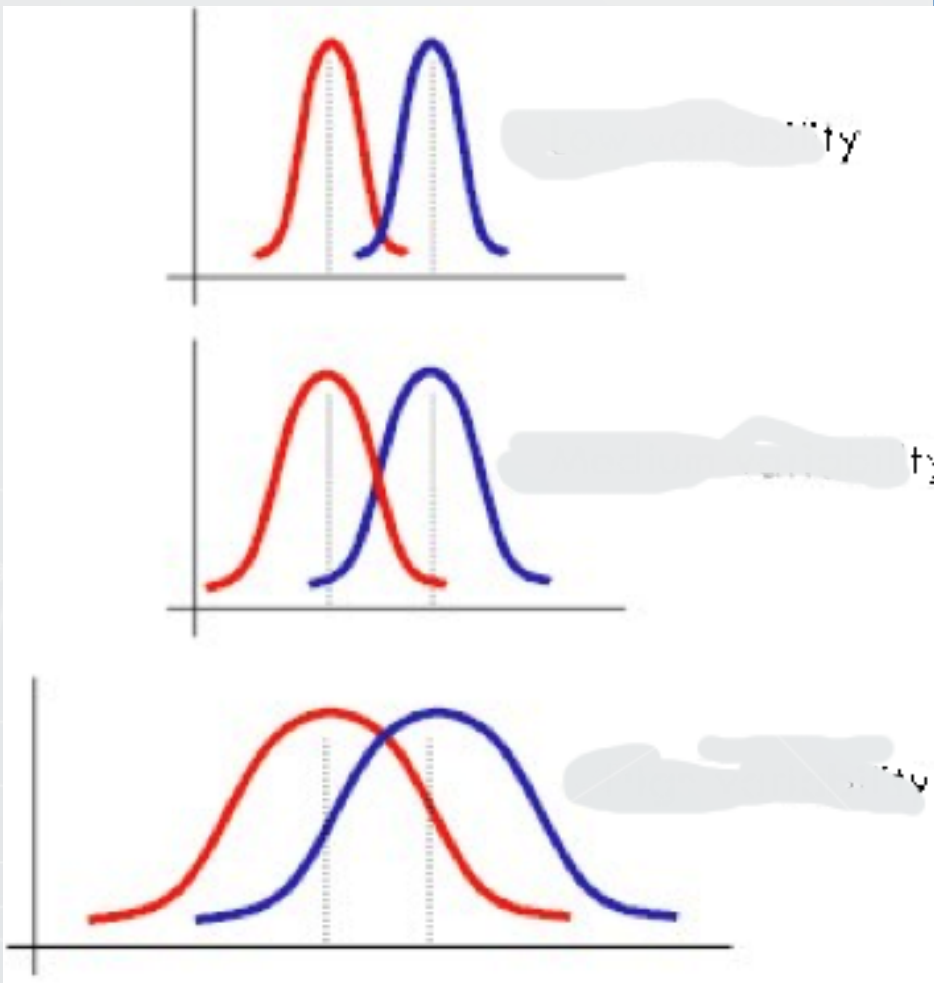
- Calculate a test statistic from the data (often assuming something about the data)
- Based on this statistic, with *some probability* we can **reject the null hypothesis**, that is, show that it does not hold

t-test

Determines whether two samples come from the same underlying distribution or not



?



t-test

Null hypothesis: model 1 and model 2 accuracies are no different, i.e. come from **the same** distribution

Result: probability that the difference in accuracies is due to random chance (low values are better)

Calculating t-test

For our setup, we'll do what's called a "pair t-test"

- The values can be thought of as pairs, where they were calculated under the same conditions
- In our case, the same train/test split
- Gives more power than the unpaired t-test (we have more information)

For almost all experiments, we'll do a "two-tailed" version of the t-test

http://en.wikipedia.org/wiki/Student's_t-test

p-value

- The result of a statistical test is often a p-value
- p-value: the probability that the null hypothesis holds. Specifically, if we re-ran this experiment multiple times (say on different data) what is the probability that we would reject the null hypothesis incorrectly (i.e. the probability we'd be wrong)
- Common values to consider "significant": 0.05 (95% confident), 0.01 (99% confident) and 0.001 (99.9% confident)

Comparing systems: sample 1

split	model 1	model 2
1	87	88
2	85	84
3	83	84
4	80	79
5	88	89
6	85	85
7	83	81
8	87	86
9	88	89
10	84	85
average:	**85**	**85**

Is model 2 better than model 1?

They are the same with: p = 1

Comparing systems: sample 2

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
average:	**82**	**85**

Is model 2 better than model 1?

They are the same with: p = 0.15

Comparing systems: sample 3

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**

Is model 2 better than model 1?

They are the same with: p = 0.007

Comparing systems: sample 4

split	model 1	model 2
1	80	82
2	84	87
3	89	90
4	78	82
5	90	91
6	81	83
7	80	80
8	88	89
9	76	77
10	86	88
average:	**83**	**85**

Is model 2 better than model 1?

They are the same with: p = 0.001

Statistical tests on test data

Labeled Data

(data with labels)

All Training Data

Training Data

Development Data

cross-validation with t-test

Test Data

Can we do that here?

Bootstrap resampling

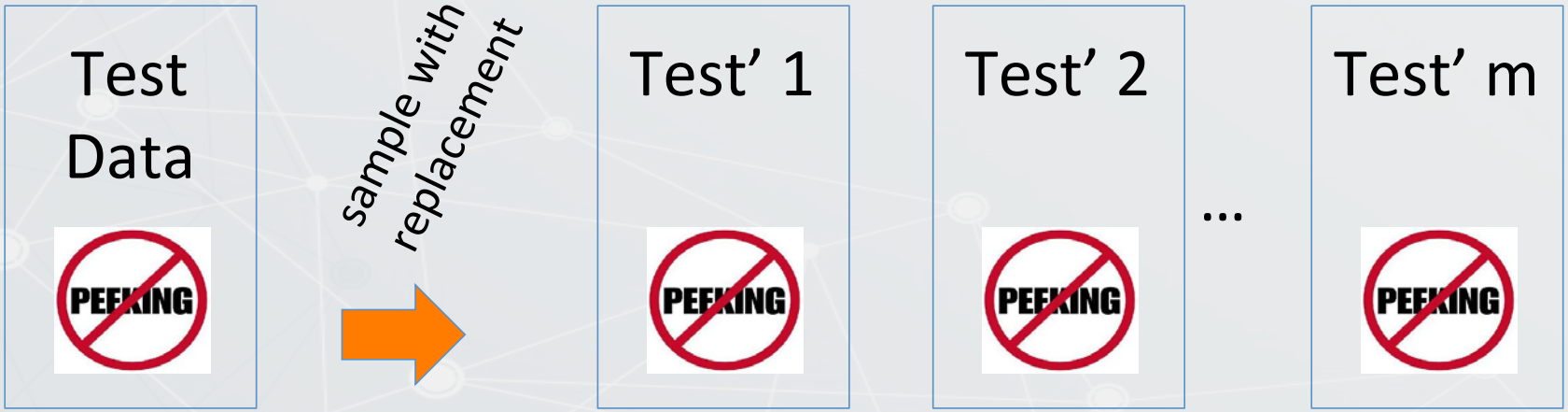
test set *t* with n samples

do *m* times:

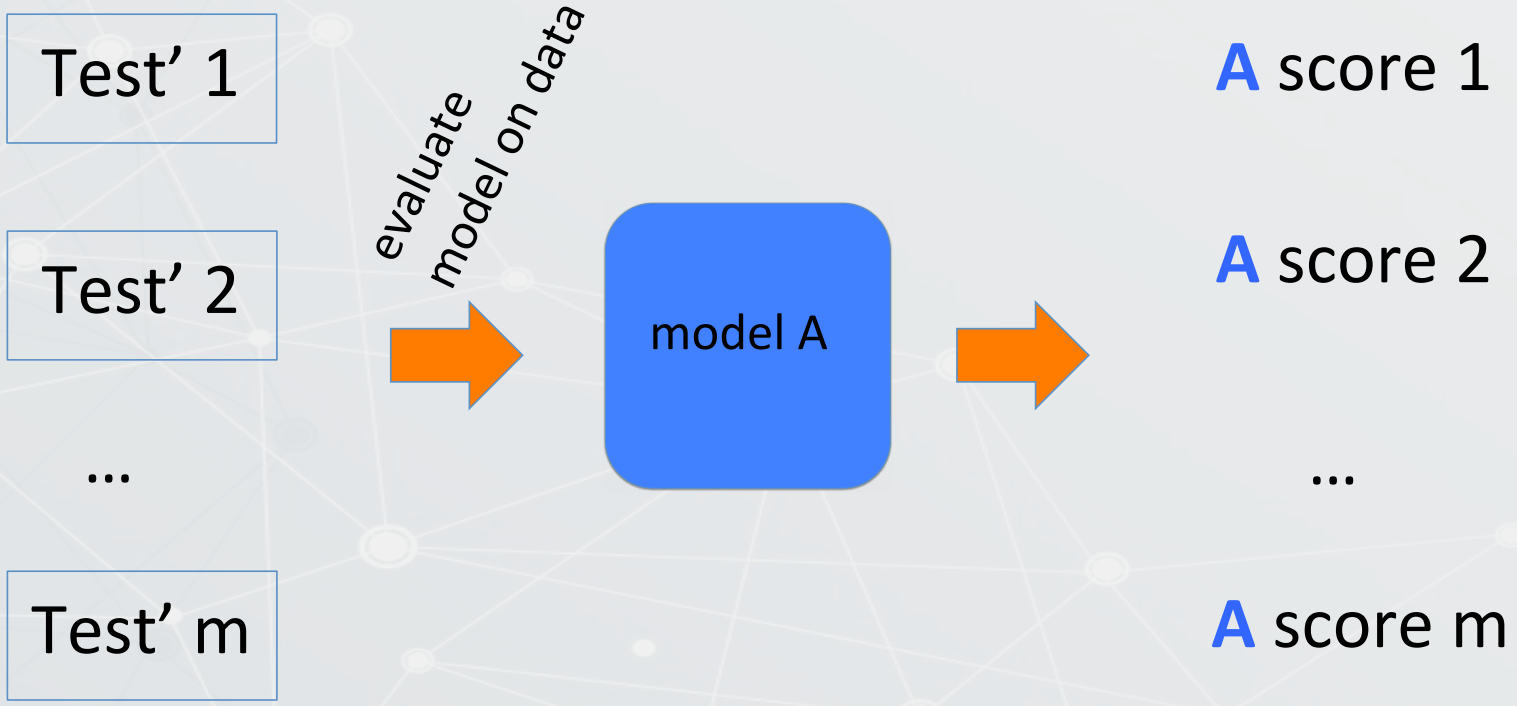
- sample *n* examples **with replacement** from the test set to create a new test set *t'*
- evaluate model(s) on *t'*

calculate t-test (or other statistical test) on the collection of *m* results

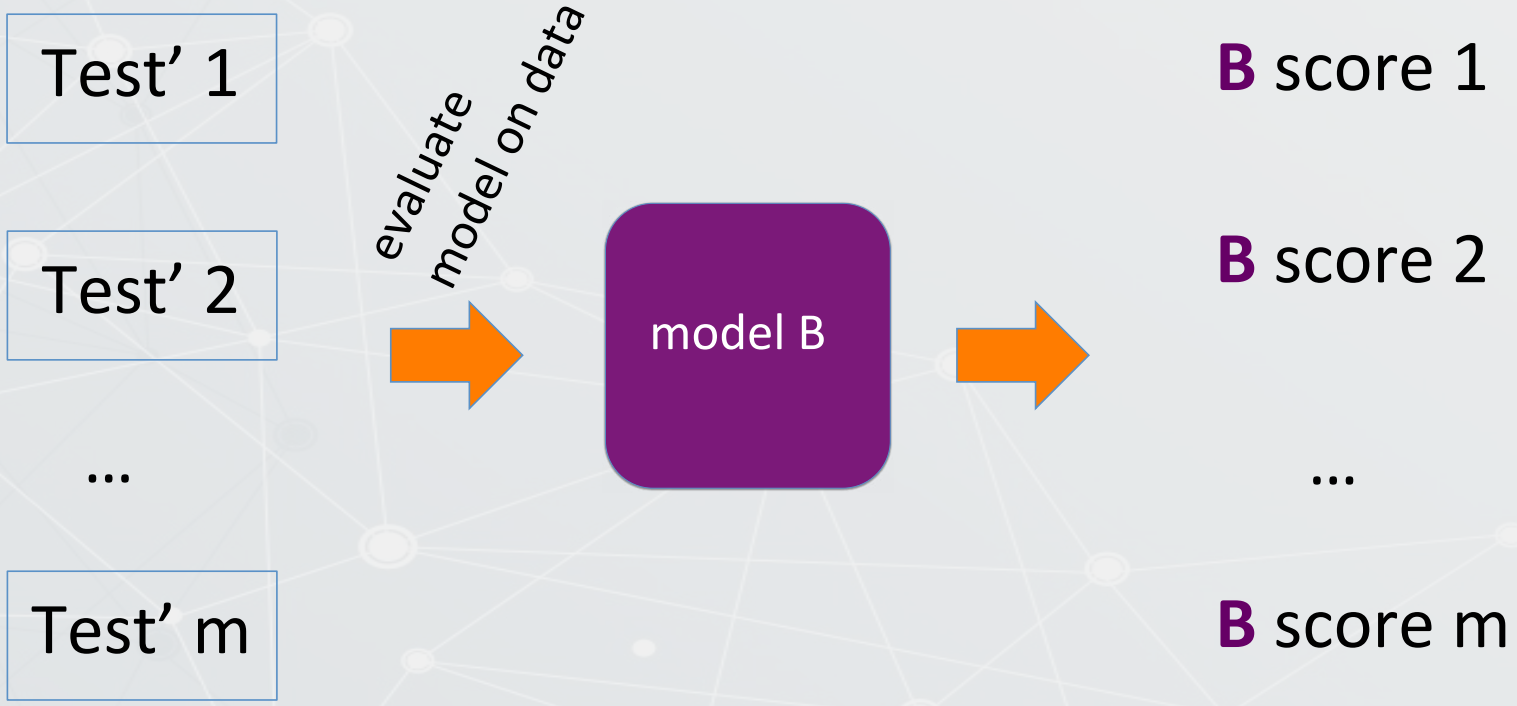
Bootstrap resampling



Bootstrap resampling



Bootstrap resampling



Bootstrap resampling

A score 1	**B** score 1
A score 2	**B** score 2
…	…
A score m	**B** score m

paired t-test (or other analysis)

Experimentation good practices

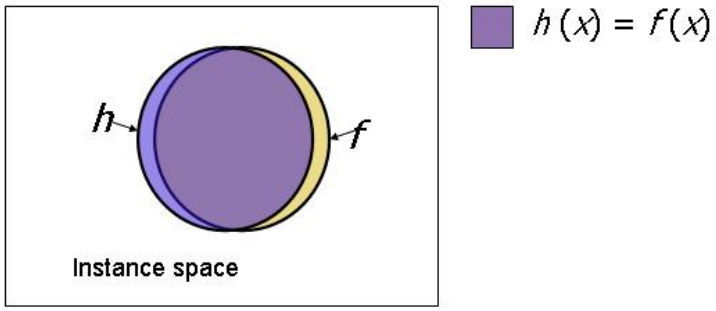
Never look at your test data!

During development

- Compare different models/hyperparameters on development data
- use cross-validation to get more consistent results
- If you want to be confident with results, use a t-test and look for p = 0.05 (or even better)

For final evaluation, use bootstrap resampling combined with a t-test to compare models

Estimating the performance of a classifier



The *true* error of a hypothesis *h* with respect to a target function *f* and an instance distribution *D* is

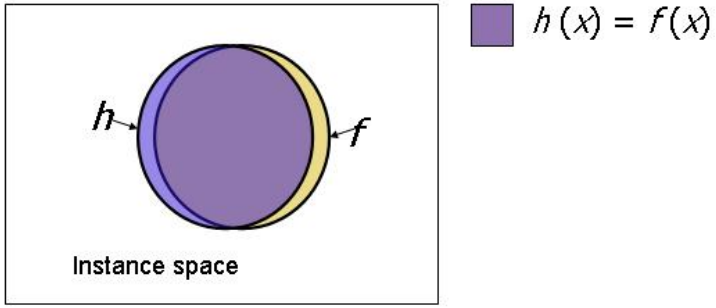
$$Error_D(h) \equiv \Pr_{x \in D}\left[f(x) \neq h(x)\right]$$

The sample error of a binary classifier *h* with respect to a target function *f* and an instance distribution D is

$$Error_S(h) \equiv \frac{1}{|S|}\sum_{x \in S} \delta(f(x) \neq h(x))$$

$$\delta(a,b) = 1 \text{ iff } a \neq b; \delta(a,b) = 0 \text{ otherwise}$$

Estimating classifier performance



$$Domain(X) = \{a, b, c, d\}$$

$$D(X) = \left\{\frac{1}{8}, \frac{1}{2}, \frac{1}{8}, \frac{1}{4}\right\}$$

x	a	b	c	d
$h(x)$	0	1	1	0
$f(x)$	1	1	0	0

$$error_D(h) = \Pr_D[h(x) \neq f(x)]$$
$$= D(X = a) + D(X = c)$$
$$= \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$$

Evaluating the performance of a classifier

- Sample error estimated from training data is an *optimistic* estimate

$$Bias = E\left[Error_S(h)\right] - Error_D(h)$$

- For an *unbiased* estimate, *h* must be evaluated on an independent sample *S* (which is not the case if *S* is the training set!)
- Even when the estimate is unbiased, it can *vary* across samples!
- If *h* misclassifies 8 out of 100 samples

$$Error_S(h) = \frac{8}{100} = 0.08$$

How close is the sample error to the true error?

How close is the *estimated* error to the *true* error?

- Choose a sample *S* of size *n* according to distribution *D*
- Measure

$$Error_S(h)$$

$Error_S(h)$ is a random variable (outcome of a random experiment)

Given $Error_S(h)$, what can we conclude about $Error_D(h)$?

More generally, given the estimated performance of a hypothesis, what can we say about its actual performance?

Evaluating performance when we can afford to test on a large independent test set

The *true* error of a hypothesis *h* with respect to a target function *f* and an instance distribution *D* is

$$Error_D(h) \equiv \Pr_{x \in D}\left[f(x) \neq h(x)\right]$$

The sample error of a classifier *h* with respect to a target function f and an instance distribution *D* is

$$Error_S(h) \equiv \frac{1}{|S|}\sum_{x \in S}\delta(f(x) \neq h(x))$$

$$\delta(a,b) = 1 \text{ iff } a \neq b; \delta(a,b) = 0 \text{ otherwise}$$

Evaluating Classifier performance

$$Bias = E\left[Error_S(h)\right] - Error_D(h)$$

Sample error estimated from training data is an *optimistic* estimate

For an *unbiased* estimate, h must be evaluated on an independent sample S (which is not the case if S is the training set!)

Even when the estimate is unbiased, it can *vary* across samples!

If *h* misclassifies 8 out of 100 samples

$$Error_S(h) = \frac{8}{100} = 0.08$$

How close is the sample error to the true error?

How close is estimated error to its true value?

Choose a sample *S* of size *n* according to distribution *D*

Measure $Error_S(h)$

$Error_S(h)$ is a random variable (outcome of a random experiment)

Given $Error_S(h)$, what can we conclude about $Error_D(h)$?

More generally, given the estimated performance of a classifier, what can we say about its actual performance?

How close is estimated accuracy to its true value?

Question: How close is p (the true probability) to $\hat{p}$?

This problem is an instance of a well-studied problem in statistics

- The problem of estimating the proportion of a population that exhibits some property, given the observed proportion over a random sample of the population.
- In our case, the property of interest is that h correctly (or incorrectly) classifies a sample.
- Testing h on a single random sample x drawn according to D amounts to performing a random experiment which succeeds if h correctly classifies x and fails otherwise.

How close is estimated accuracy to its true value?

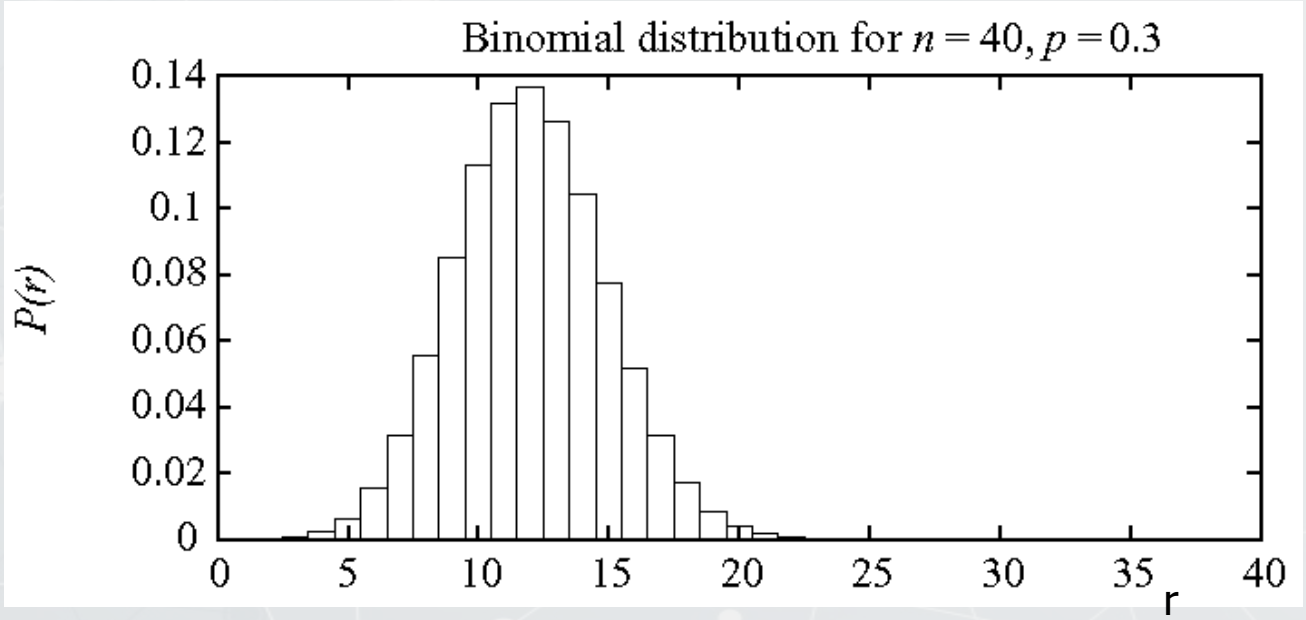
The output of a classifier whose true error is *p* as a binary *random variable* which corresponds to the outcome of a Bernoulli trial with a *success rate p* (the probability of correct prediction)

The *number* of *successes r* observed in *N* trials is a random variable Y which follows the Binomial distribution

$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

$Error_S(h)$ is a Random Variable

Probability of observing r misclassified examples in a sample of size n:



$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

Recall basic statistics

Consider a random experiment with discrete valued outcomes $y_1, y_2, \cdots y_M$

The expected value of the corresponding random variable *Y* is

$$E(Y) \equiv \sum_{i=1}^{M} y_i \Pr(Y = y_i)$$

The variance of *Y is* $Var(Y) \equiv E\left[(Y - E[Y])^2\right]$

The standard deviation of *Y* is $\sigma_Y \equiv \sqrt{Var(Y)}$

How close is estimated accuracy to its true value?

The *mean* of a Bernoulli trial with success rate $p = p$

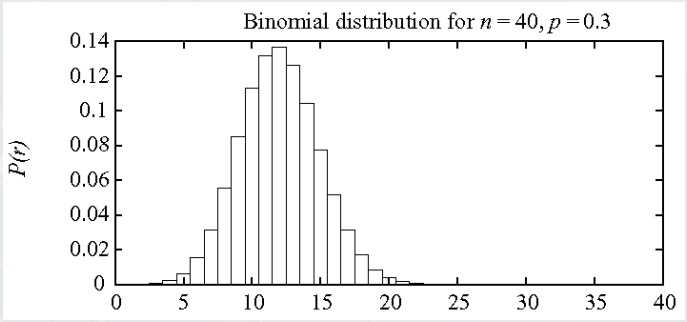
Variance = p (*1-p*)

If *N trials* are taken from the same Bernoulli process, the observed success rate $\hat{p}$ has the same mean p

and variance $\frac{p(1-p)}{N}$

For large *N*, the distribution of $\hat{p}$ follows a Gaussian distribution

Binomial Probability Distribution



$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

Probability P(r) of r heads in n coin flips, if p = Pr(heads)

- Expected, or mean value of X, E[X], is

$$E[X] \equiv \sum_{i=0}^{N} iP(i) = np$$

- Variance of X is

$$Var(X) \equiv E[(X - E[X])^2] = np(1-p)$$

- Standard deviation of X, σ_X, is

$$\sigma_X \equiv \sqrt{E[(X - E[X])^2]} = \sqrt{np(1-p)}$$

Estimators, Bias, Variance, Confidence Interval

$$Error_S(h) = \frac{r}{n}$$

$$Error_D(h) = p$$

$$\sigma_{Error_S(h)} = \sqrt{\frac{p(1-p)}{n}}$$

$$\sigma_{Error_S(h)} = \sqrt{\frac{Error_D(h)(1-Error_D(h))}{n}}$$

$$\sigma_{Error_S(h)} \approx \sqrt{\frac{Error_S(h)(1-Error_S(h))}{n}}$$

An N% confidence interval for some parameter p that is the interval which is expected with probability N% to contain p

Normal distribution approximates binomial

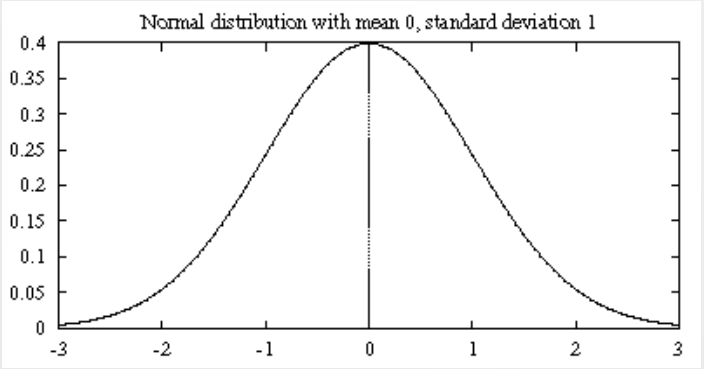
$Error_S(h)$ follows a Binomial distribution, with

- mean $\mu_{Error_S(h)} = Error_D(h)$
- standard deviation $\sigma_{Errors_S(h)} = \sqrt{\frac{Error_D(h)(1-Errors_D(h))}{n}}$

We can approximate this by a Normal distribution with the same mean and variance when $np(1-p) \geq 5$

Normal distribution

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$



The probability that X will fall in the interval (a, b) is given by $\int_a^b p(x)dx$

Expected, or mean value of X is given by E[X] = μ

Variance of X is given by Var(X) = σ^2

Standard deviation of X is given by $\sigma_X = \sigma$

How close is the estimated accuracy to its true value?

Let the probability that a Gaussian random variable X, with zero mean, takes a value between –z and z,

$Pr[-z \leq X \leq z] = c$

Pr[X ≥ z]	z
0.001	3.09
0.005	2.58
0.01	2.33
0.05	1.65
0.10	1.28

How close is the estimated accuracy to its true value?

But $\hat{p}$ does not have zero mean and unit variance so we normalize to get

$$\Pr\left[-z < \frac{\hat{p} - p}{\sqrt{\frac{p(1-p)}{n}}} < z\right] = c$$

How close is the estimated accuracy to its true value?

To find confidence limits:

Given a particular confidence figure *c,* use the table to find the z corresponding to the probability ½ (*1-c*).

Use linear interpolation for values not in the table

$$p = \frac{\left[\hat{p} + \frac{z^2}{2n} \pm z\sqrt{\frac{\hat{p}}{n} - \frac{\hat{p}^2}{n} + \frac{z^2}{4n^2}}\right]}{\left[1 + \frac{z^2}{n}\right]}$$