Performance Evaluation and Hypothesis Testing

Motivation

Evaluating the performance of learning systems is important because:

- Learning systems are usually designed to predict the class/value of "future" unlabeled data points
- In some cases, evaluating *alternative models* (that we call «hypotheses») is an integral part of the learning process
- For example, in neural networks, different network architectures – with different numbers of hidden layers –represent alternative hypotheses.

Which one is the best predictor of reality?

The «real» function and the hypothesis

Whether our algorithm must learn a discrete *c(x)* or continuous *f(x)* function ,the problem is that we are given the «true» values of the function ONLY for the points (examples) of the training set D

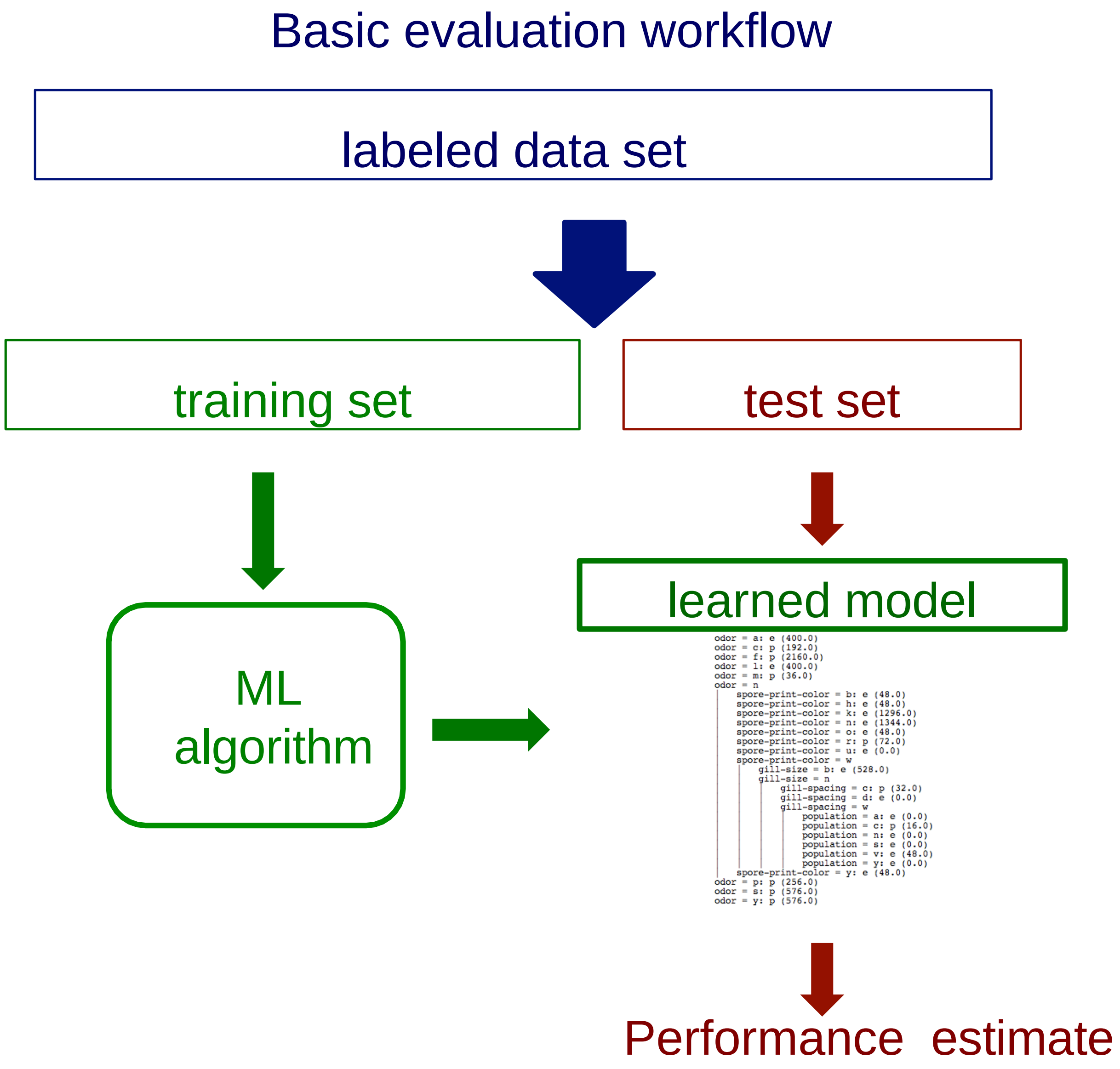
Learning a model amounts to learning a function h(x) – named an *hypothesis* – that approximates the unknown function at best (*note we now use h(x) rather than f(x) or c(x), to highlight the fact that ML systems learn* ***approximate*** *solutions of a given problem!*)

Perfect learning is not possible in the majority of real-life cases

During the learning process, ML algorithms try to «fit» at best h(x) (usually, in an iterative manner) on training data so as to minimize errors on the training set points

Once an hypothesis is learned, **we must evaluate its quality**

Evaluation in (supervised) ML systems



How to select a training set? Stratified sampling

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

labeled data set

++++++++++++ - - - - - - - -

↓

training set

++++++ - - - -

test set

++++++ - - - -

↓

validation set

+++ - -

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

How to monitor the training process?

Learning curves:

A learning curve is a plot showing the progress in terms of performance (the Loss, or any chosen performance measure) w.r.t. a specific metrics related to learning, during the training of a machine learning model.

Learning curves by training set size

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

this can be assessed by plotting *learning curves*

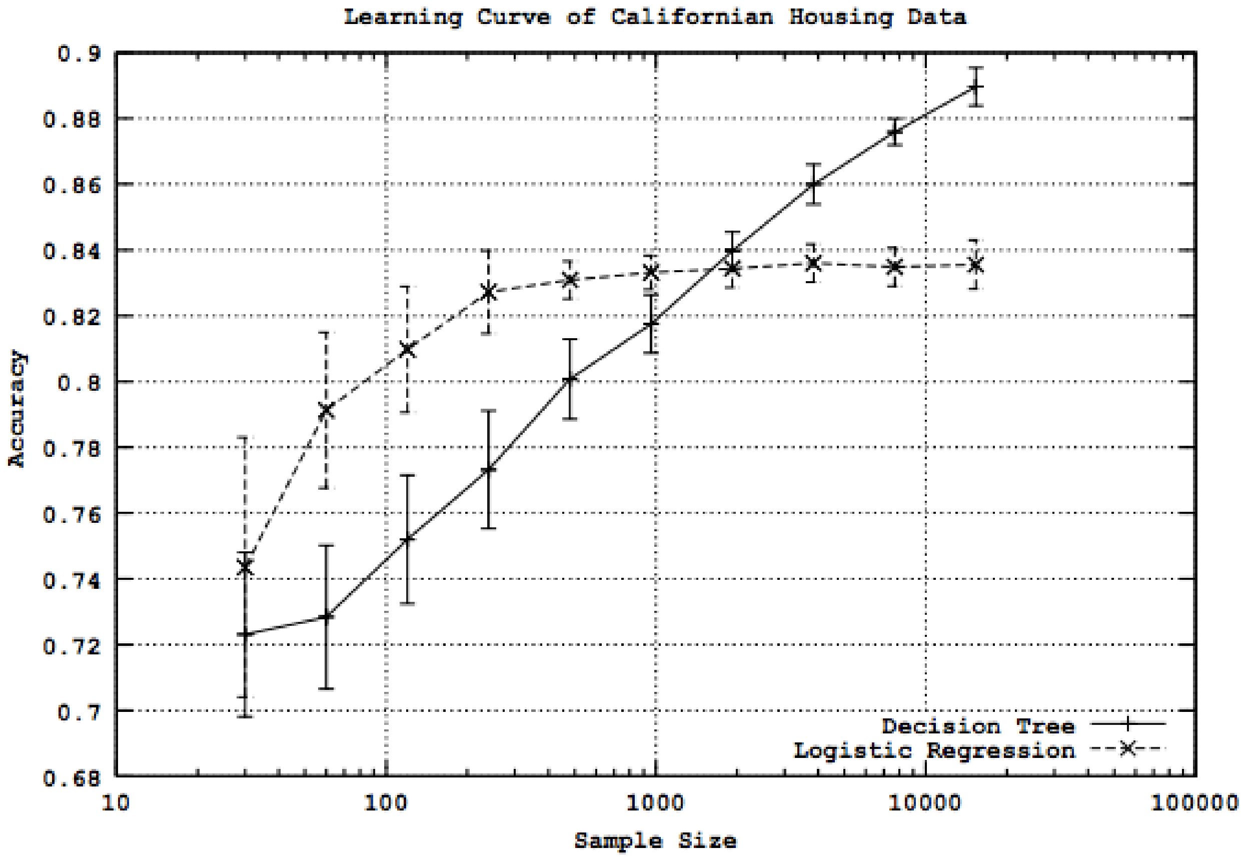
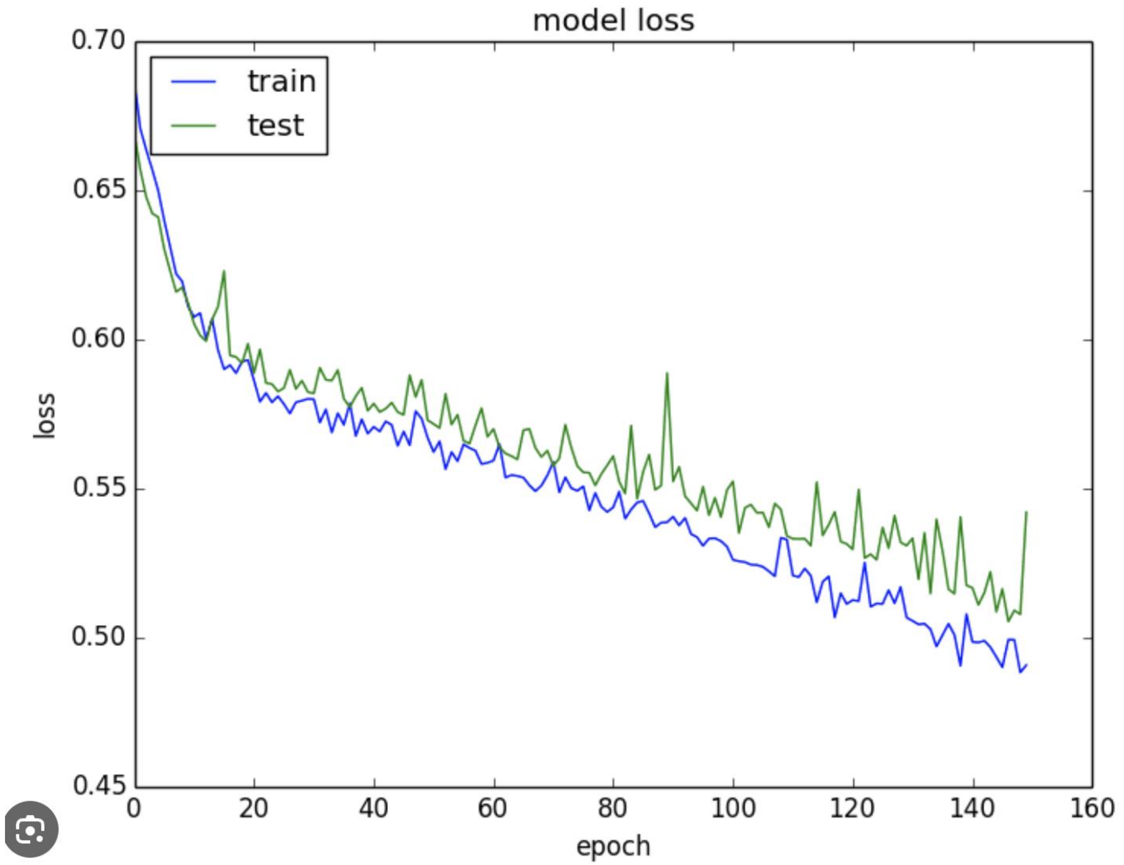
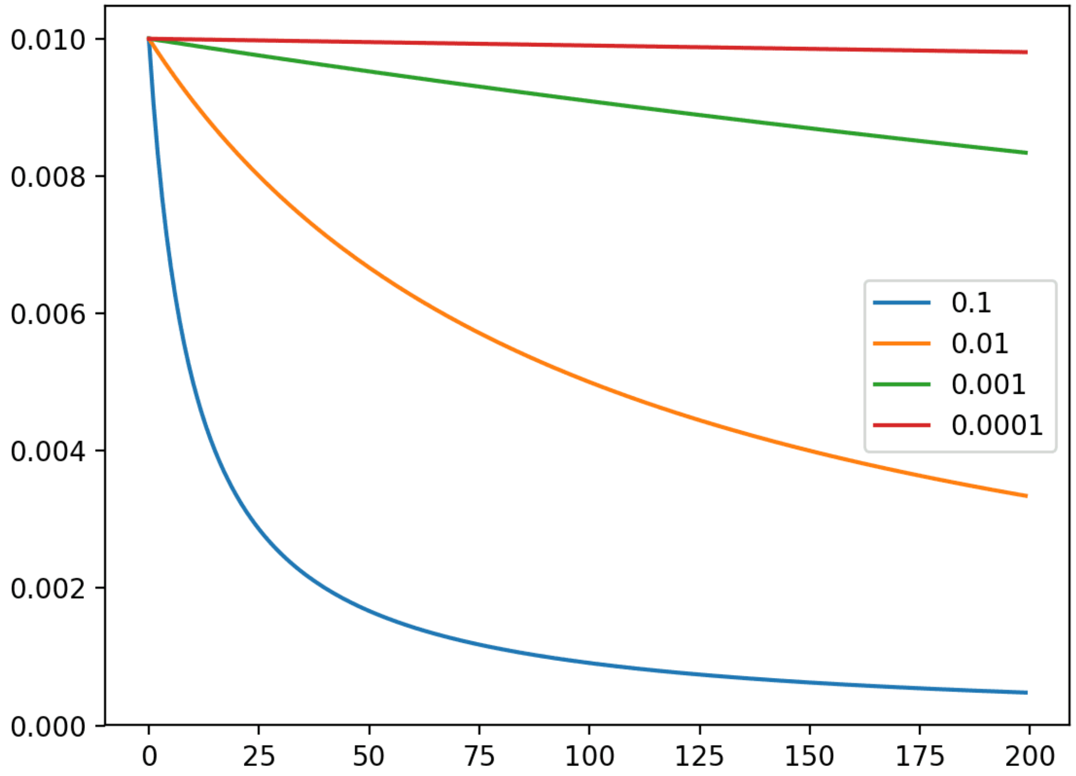


Figure from Perlich et al. *Journal of Machine Learning Research*, 2003



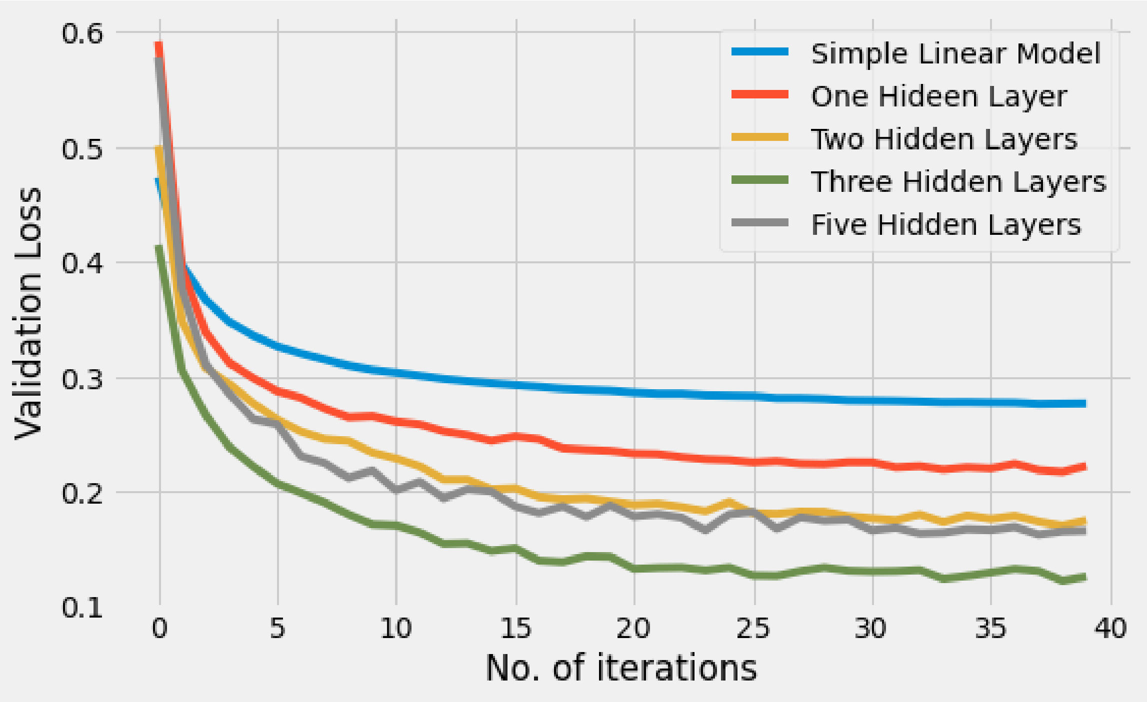
Learning curves by n. of epochs

Learning curves by learning rate η



Many types of learning curves can be plotted according to different settings of the hypeparameters

Learning curve by n. of hidden layers



Issues in performance evaluation

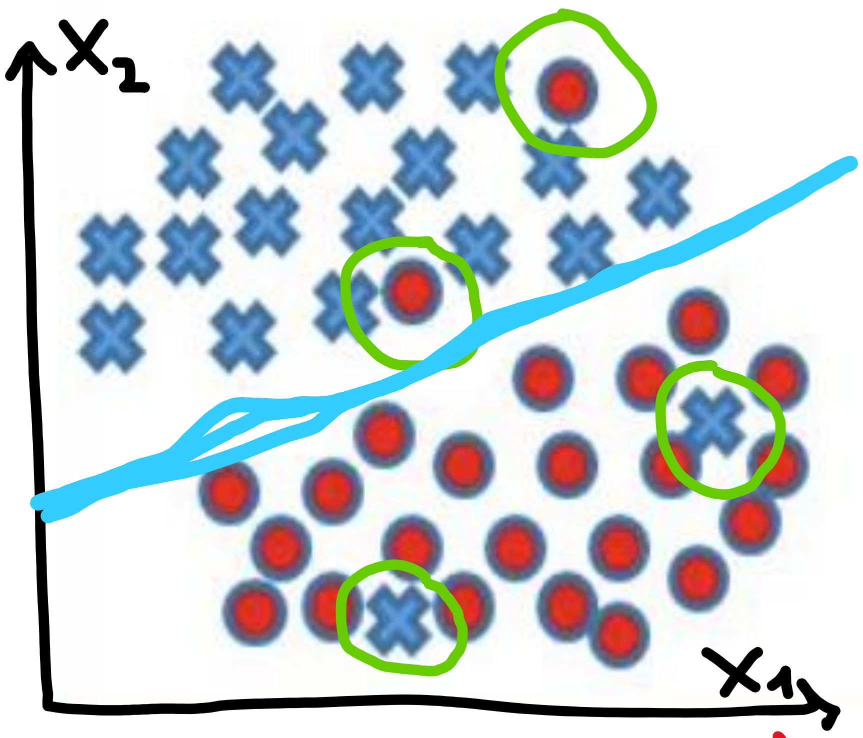
1. Which performance measure we should use?
2. How well can a classifier be expected to perform on “novel” data, not used for training?
3. Since a performance measure is an estimate on a sample, how accurate is our estimate?
4. How to compare performances of different hypotheses or those of different classifiers?

Which performance shoud we use?

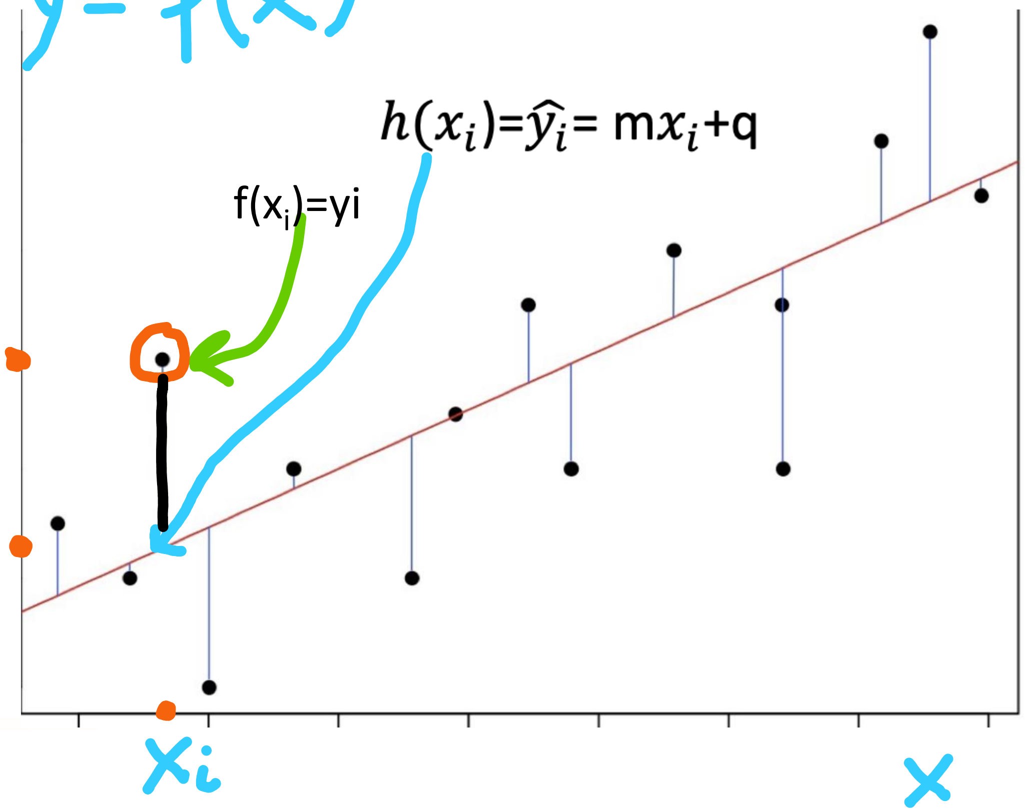
- Performance measures are a function of the **errors** made by the current model
- Adopted performance measures depend on whether we are learning a classifier or a regressor
- For <u>classifiers</u>, e.g., perceptron, the error function is binary: either the learned model is correct (it predicts the right class) or it is wrong
- For <u>regressors</u>, we must take into account the «distance» between the predicted value and the ground-truth

Classifier and (linear) regressor errors

$c(x_{1i}, x_{2i}) = Y$



$y = f(x)$



e

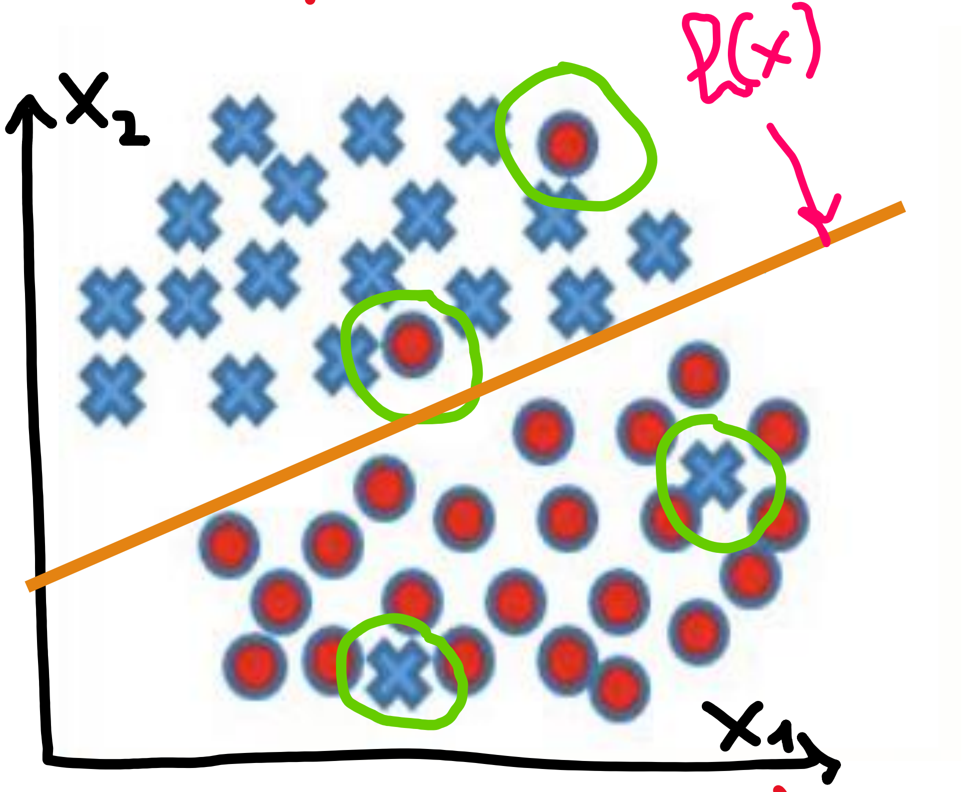
$Y = \{\text{red}, \text{blue}\}$

$\quad\ \ 0 \qquad\quad 1$

Performance measures for classifiers

Classifier error (measured on the test set)

$c(x_{1i}, x_{2i}) = Y$



e $Y = \{red, blue\}$

$y_i = c(\mathbf{x}_i)$ is the correct classification $\hat{y}_i = h(\mathbf{x}_i)$ is the output of the classifier

$$error(h(x)) = \sum_{i=1}^{n} \delta(c(\boldsymbol{x_i}), h(\boldsymbol{x_i}))$$

δ(x,y)=0 if x=y, else δ(x,y)=1

δ It is called the **Kronecker** function

Performances of classifiers

- For classifiers, often it matters to distinguish the **types** of errors: is the system misclassifying the «reds» or the «blues»??
- Performances are usually reported in the form of a confusion matrix (also called **contingency table**)
- The table has four cells (in case of binary classifiers):

➢ **True Positive** (TP): number of positive (=blue, =1..) instances classified as positive by the system

➢ **True Negative** (TN): number of negative (=red, =0,..) instances classified as negative by the system

➢ **False Positive** (FP): number of negative instances classified as positive by the system

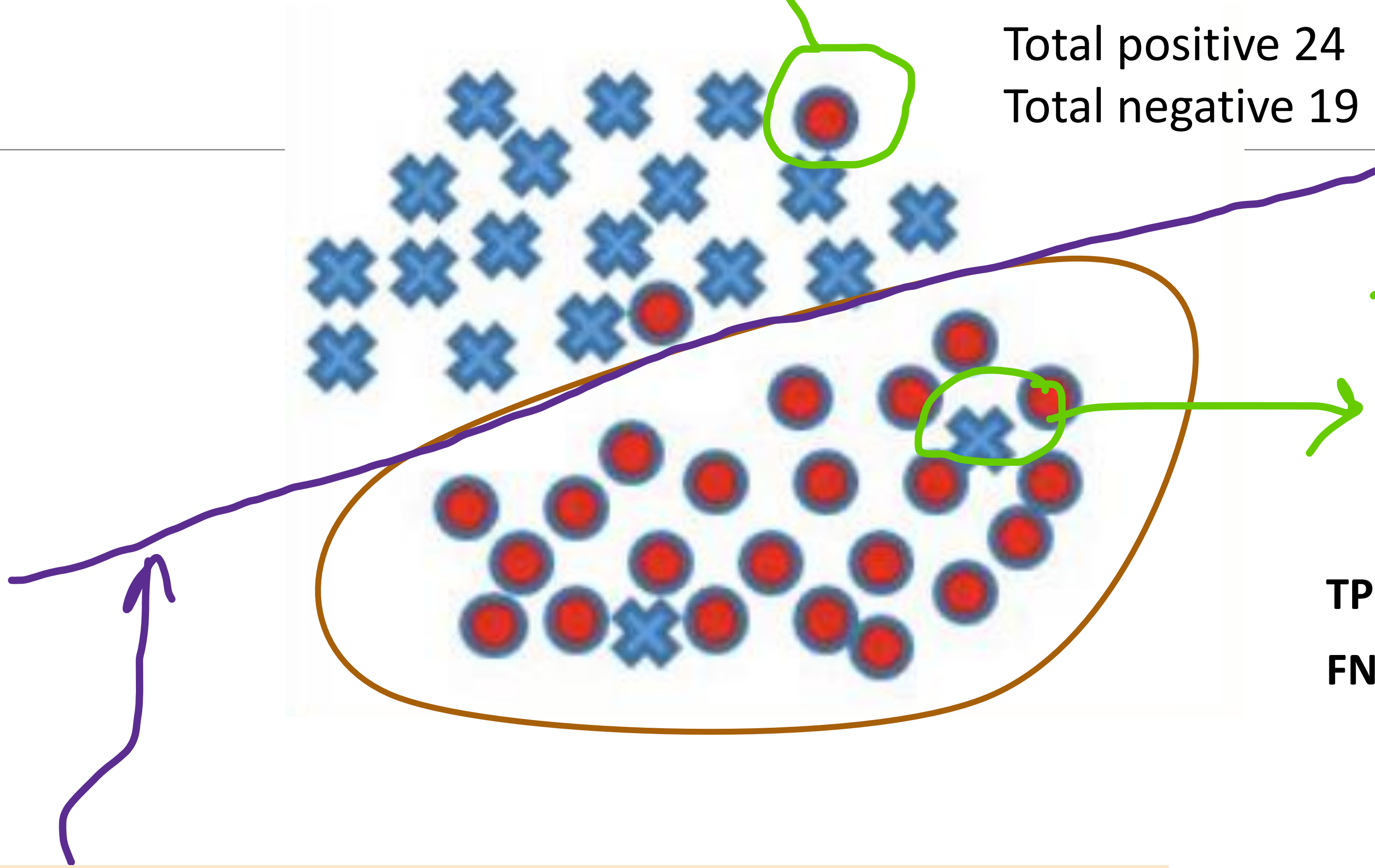
➢ **False Negative** (FN): number of positive instances classified as negative by the system

Contingency Table

		ACTUAL CLASS	
		POSITIVE	NEGATIVE
PREDITED CLASS	POSITIVE	TRUE POSITIVE (TP)	FALSE POSITIVE (FP)
	NEGATIVE	FALSE NEGATIVE (FN)	TRUE NEGATIVE (TN)

FN

Total instances 43
Total positive 24
Total negative 19



FP

TP: 22

FN: 2

FP: 2

TN: 17

Learned decision boundary is f(x)
If f(x)>0 then c(x)=0 (negative) else c(x)=1 (positive)

Performances measures of classifiers (1)

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

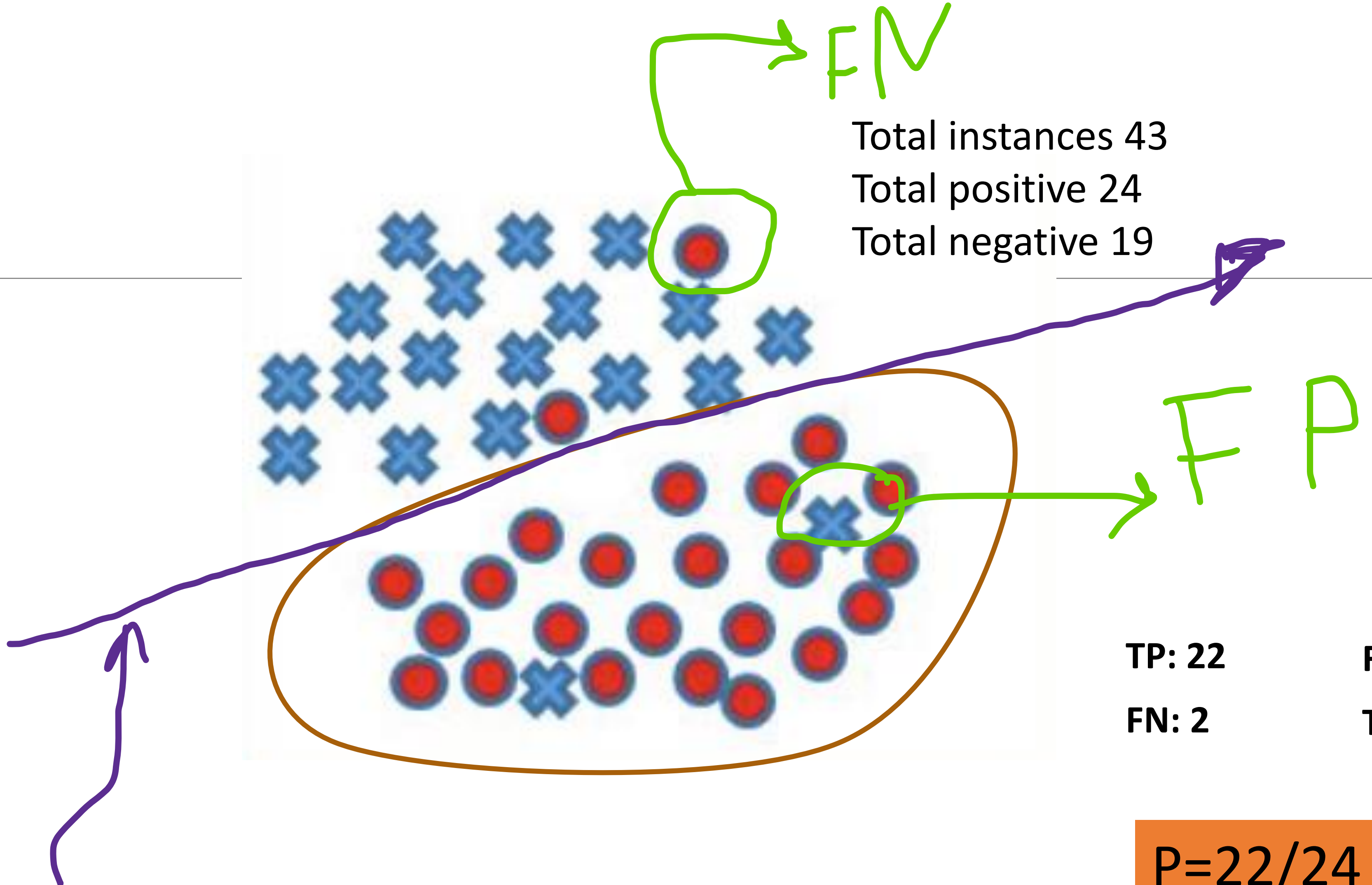
Also called **Sensitivity or True Positive rate**

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

$$ACCURACY = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\boxed{Error\ rate = Classification\ error = \frac{FP+FN}{TP + TN + FP + FN} = 1 - Accuracy}$$

Total instances 43
Total positive 24
Total negative 19



TP: 22	**FP: 2**
FN: 2	**TN: 17**

Learned decision boundary is f(x)
If f(x)>0 then c(x)=0 (negative) else c(x)=1 (positive)

P=22/24 R=22/24
A= (22+17)/43

Extending to multiple classes (macro P and R)

Example: classifying future trends of a stock as UP, DOWN, STAY

The model is predicting 19 "up" but actually only 8 are truly "up"

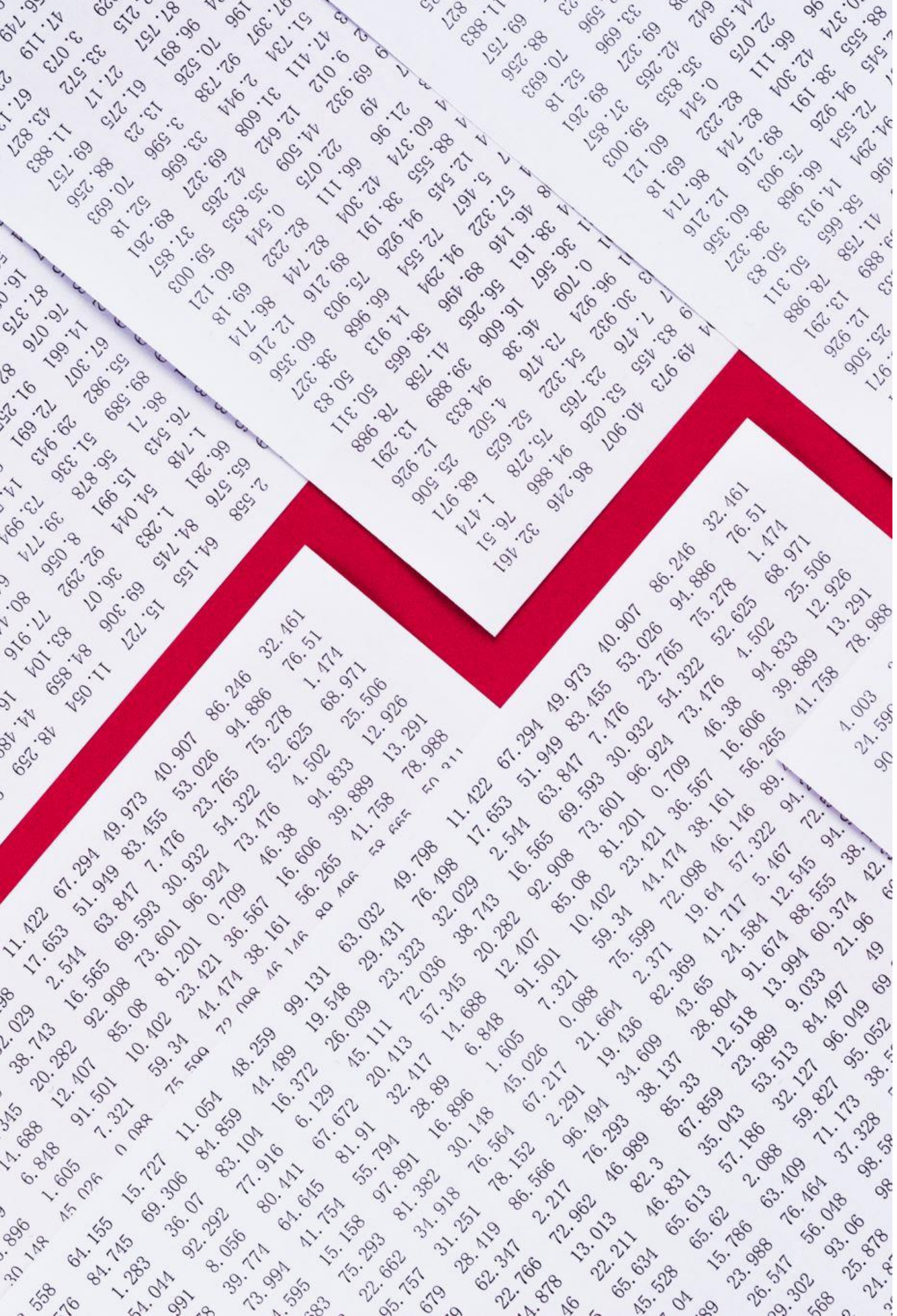
True labels

Predicted labels	up	down	stay	
up	8	10	1	$\text{precision}_u = \frac{8}{8+10+1}$
down	5	60	50	$\text{precision}_d = \frac{60}{5+60+50}$
stay	3	30	200	$\text{precision}_s = \frac{200}{3+30+200}$
	$\text{recall}_u = \frac{8}{8+5+3}$	$\text{recall}_d = \frac{60}{10+60+30}$	$\text{recall}_s = \frac{200}{1+50+200}$	

$$macro_p = \frac{precision_u + precision_d + precision_s}{nclass}$$

$$= \frac{0.42 + 0.52 + 0.86}{3}$$

$$macro_r = \frac{recall_u + recall_d + recall_s}{nclass} = \frac{0.5 + 0.6 + 0.8}{3}$$



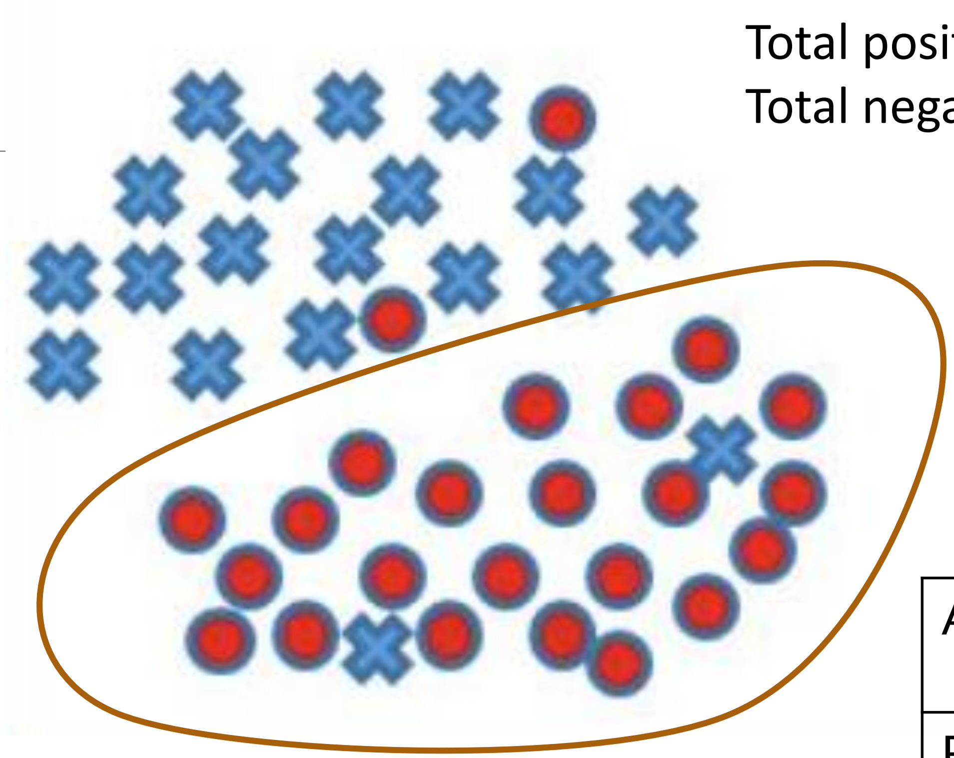
Other measures

- **Specificity** (or True Negative Rate) TN/(TN+FP) *detected negative over all negative*
- **False Positive Rate** FP/(FP+TN) *misclassified negative over all negative*
- **False Negative Rate** (Miss rate) FN/(FN+TP) *misclassified positive over all positive*

Note that FP+TN = total number of negative in test set

FN+TP = total number of positive in test set

Total instances 43
Total positive 24
Total negative 19

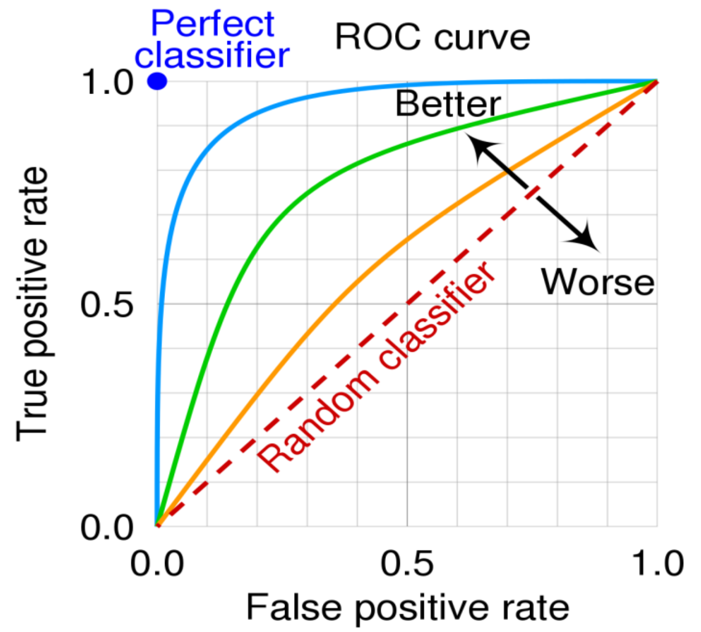


TP= 22	FP= 2
FN= 2	TN= 17

Accuracy	(22+17)/43 =0.907
Precison	22/24 =0.91
Recall (TPR)	22/24=0.91
FScore	0.91
FPR	2/19 =0,10

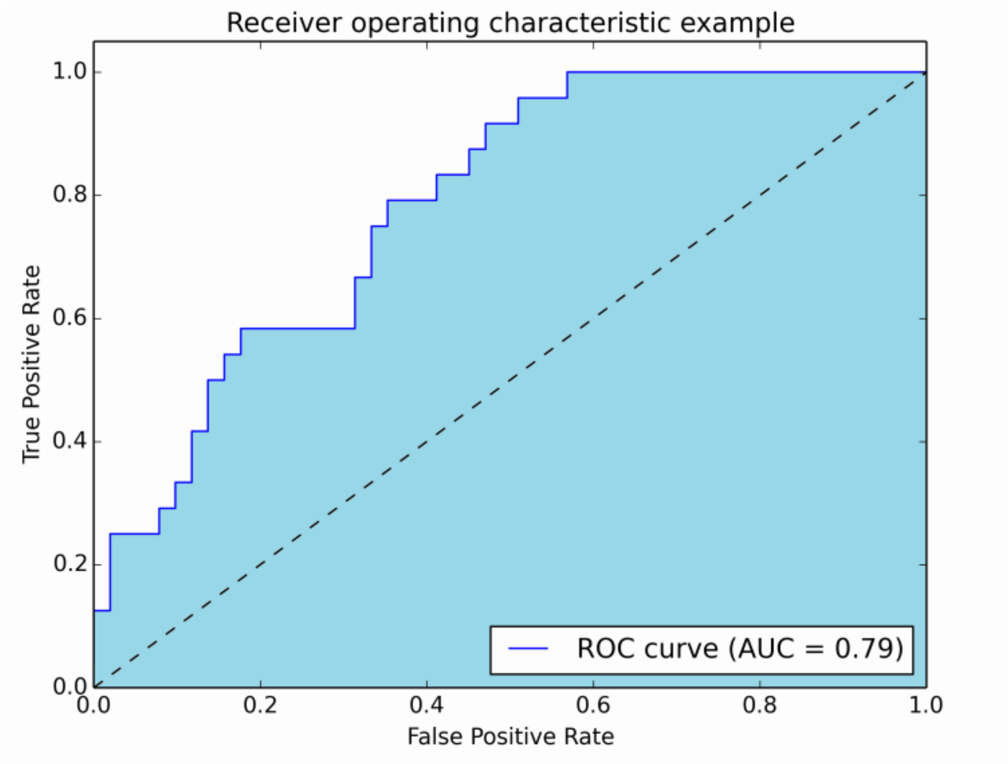
Performances measures of classifiers (2)

- **Receiver Operating Characteristic curve** (or ROC curve.) is a **graphical plot** that illustrates the performance of a **binary classifier** systems.
- The curves are created by plotting the **recall** (True Positive rate; TPR) against the **false positive rate** (FPR) at various system settings (e.g.,different hyperparameters, growing dimension of training set, etc). One **would** aim at **high recall** and **low FPR**.
- FPR=False positives/All negatives



Note that «random» is a bisector if data have an equal probability of being positive or negative

Performance measures of classifiers (3): AUROC/AUC



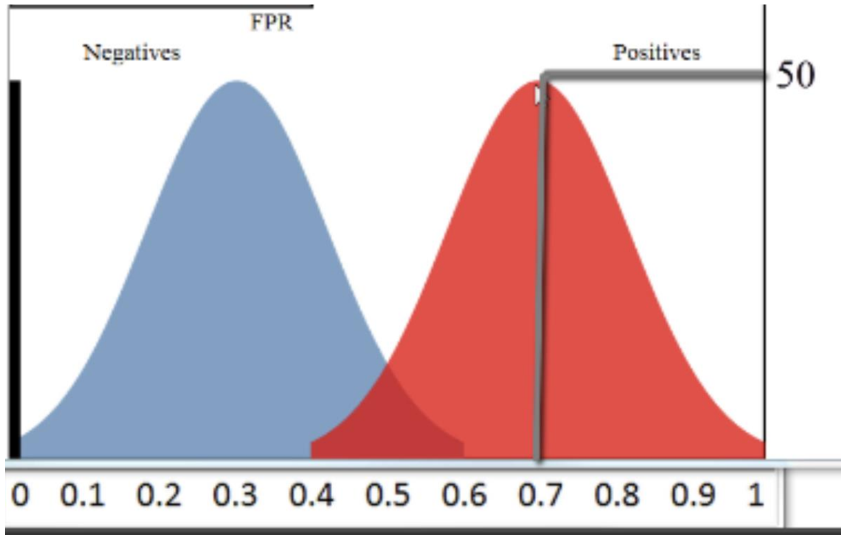
AUROC is the integral of the ROC curve

- The **Area Under the ROC (AUROC or AUC)** ROC is a probability curve and AUC represents a measure of «separability». It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.

Why is AUROC useful?

It may help understand what is the «uncertainty» zone of your predictor, and output a classification **only if outside this zone**

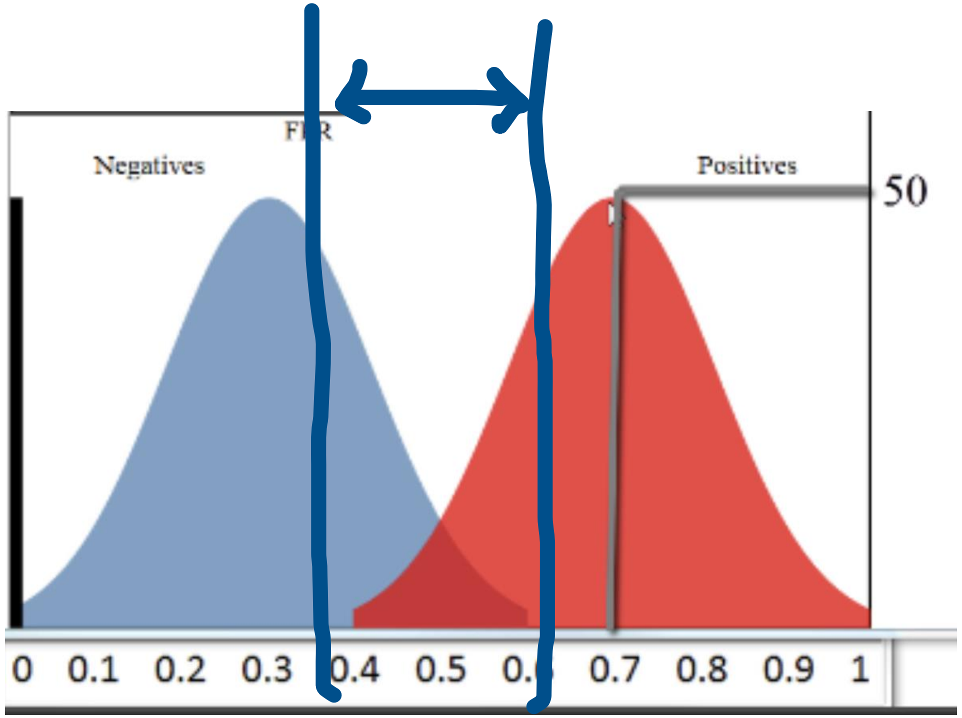
Example: we predict if a paper will be accepted at a conference, based on features like lenght of the paper, number of authors… Say red are accepted papers, blue are rejected.



Let's say our predictor output a probability, or confidence value (on the **x axis**), that a paper is accepted or not. The y axis is the **count** of observations in the **test set** (say we have 250 accepted, 250 rejected in our test set).
For example, there are 50 papers for which the system predicts p=0.7 of being positive, and they are indeed all positive.
20 papers have p=0.5, of which 10 are positive, 10 are negative

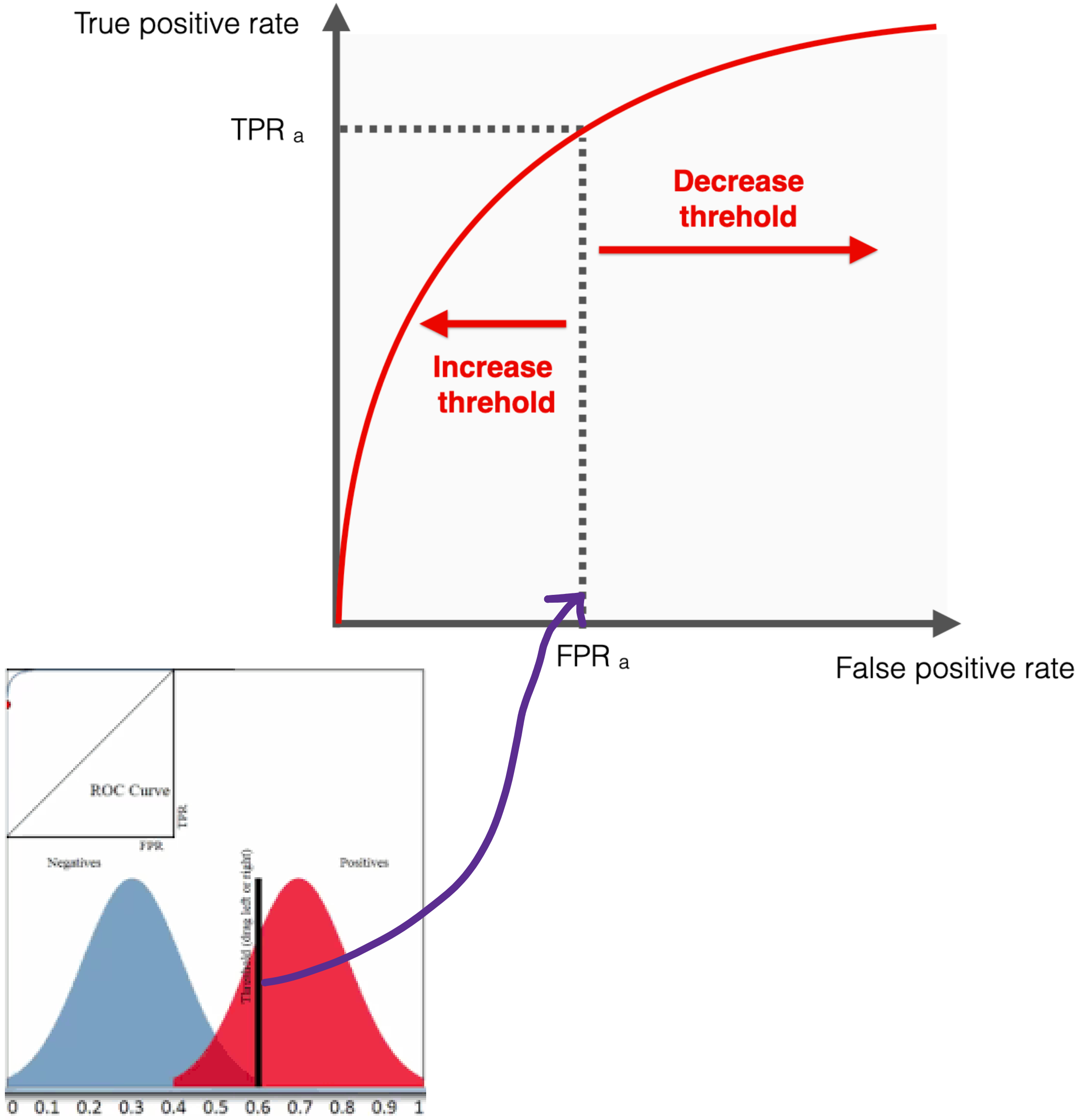
As shown in the figure, when the system output a probability between 0.4 and 0.6, it has a 50% chance of being wrong! So we should not rely on system's predictions for these output values.

AUC measures this «uncertainty area»



This is also helpful to set the "cutoff" or THRESHOLD of certain classifiers. Remember: the output of a NN is a continuous value (or a probabiity if we use softmax). The cutoff is the value above wich we predict "1" and below which we predict "0"

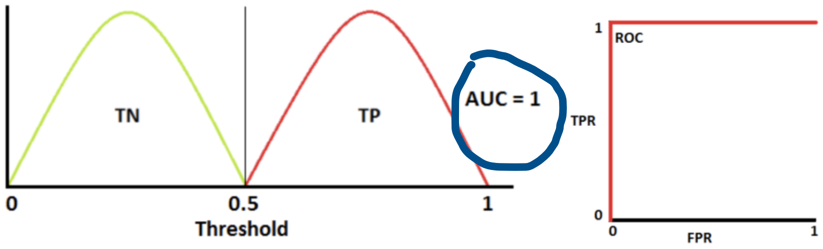
For p>0.6 the system reliably classifies positive, for p<0.4 it reliably classifies negative, in between the system is unable to correctly sperate positive from negatives



How dos this relate to ROC and AUROC?

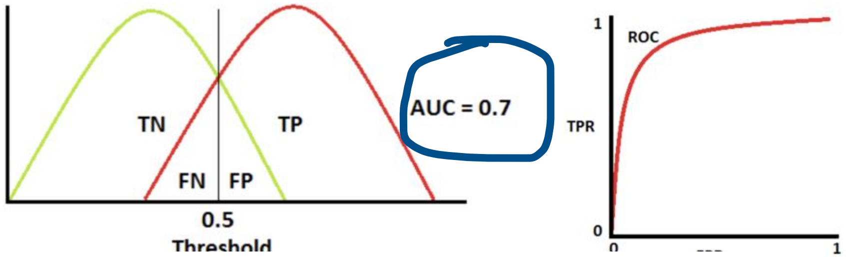
The **left side** of the ROC curve corresponds to the more "confident" thresholds: a higher threshold leads to lower recall (TPR) and fewer false positive errors. The extreme point is when both recall and FPR are 0. In this case, there are no correct detections but also no false ones.

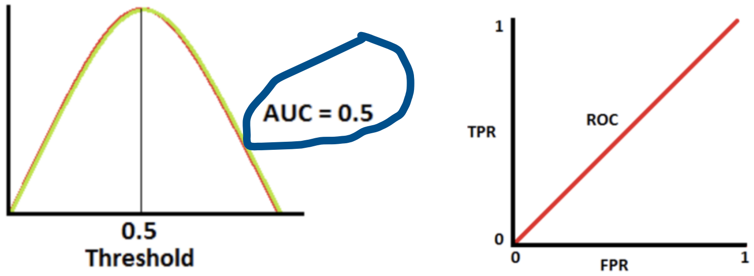
The **right side** of the curve represents the "less strict" scenarios when the threshold is low. Both recall and False Positive rates are higher, ultimately reaching 100%. If you put the threshold at 0, the model will always predict a positive class: both recall, and the FPR will be 1.

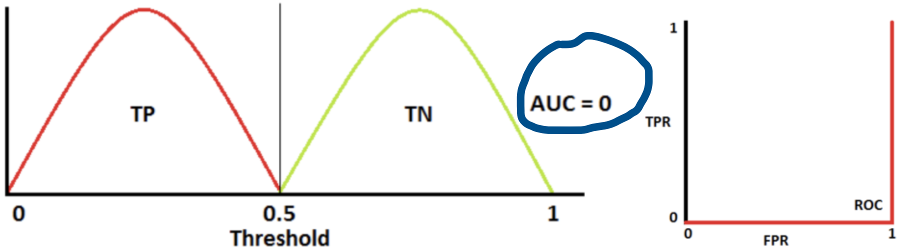


Here, TPR=1 and FPR=0, so
We have a perfect classifier

Area Under the ROC curve: the highest the value, the smallest the uncertainty zone







To summarise:

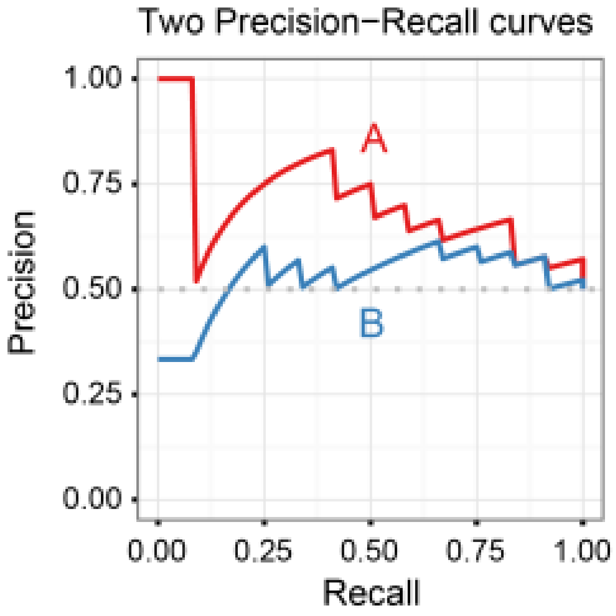
- green and red curves represent the probability that a given model classifies an instance as positive or negative given the values of its features (note: in most cases probability curves are not "nice" gaussians.. This is only an example)
- AUROC (the rightmost curves) tells us how good the model is at separating.
- More here

🡐Here the model is reciprocating the classes!

Performances measures of classifiers (4)
PR curve and AUPR

Precision-Recall curve (or PR curve) is a **graphical plot** that illustrates the performance of a **binary classifier** system. The curve is created by plotting the **recall** (True Positive rate; TPR) against the **precision** at various system settings (for example, different thresholds of a NN output such that if y≥β then c(x)=positive else negative; different hyperparameter settings, etc.).

- The **Area Under the Precision-Recall curve (AUPR)** has an intuitive meaning just like **AUROC**. However:
  - ➢ **AUROC** is better for a **binary balanced** problem.
  - ➢ **AUPR** is better for a **binary imbalanced** problem (we discussed about imbalanced classes under the topic feature engineering). See link

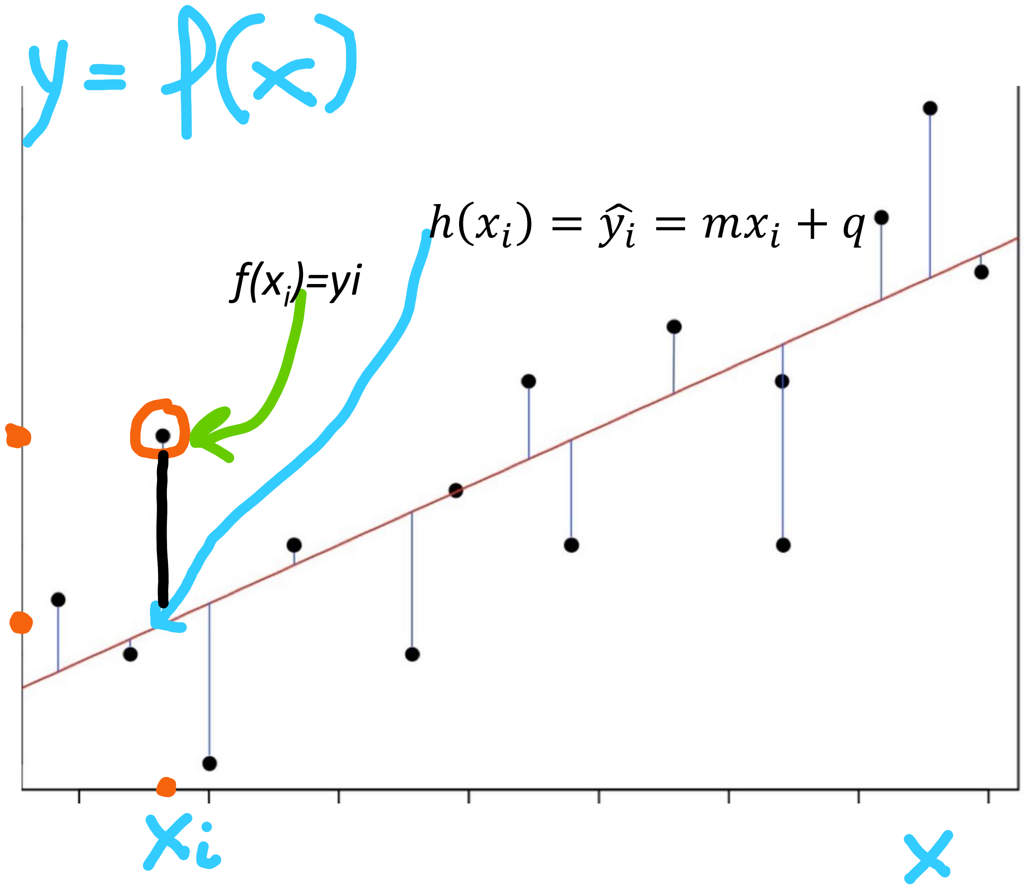


Why not ROC/AUROC with unbalanced classes?

- Suppose we have imbalanced data, e.g., in credit risk prediction, the vast majority of instances in the dataset are negative (not fraudulent users) and only a minority is positive. **We really care about capturing positive instances.**
- ROC curve is not a good visual illustration for highly imbalanced data, because the False Positive Rate (FPR=FP / (FP+TN)) **does not drop drastically when the total number of real negatives is huge** (since now FP<<TN).
- Whereas Precision (True Positives / (True Positives + False Positives)) **is highly sensitive to False Positives** and is not impacted by a large total true negative denominator.

Performance of regressors

Performance measures of Regressors



$error(h(x_i)) = f(x_i) - h(x_i)$

MAE (mean absolute error)=

$\frac{1}{n}\sum_{i=1}^{n} |f(x_i) - h(x_i)|$

RMSE Root Mean Squared Error

$$\sqrt{\frac{1}{n}\left(\sum_{i=1}^{n}(f(x_i) - h(x_i))^2\right.}$$

RSS Residual Sum of Squares

$$\left(\sum_{i=1}^{n}(f(x_i) - h(x_i))^2\right.$$

..plus many others algorithm-dependent Loss functions

Issues

1. Which performance measure we should use?
2. How well can a classifier be expected to perform on “novel” data, not used for training?
3. Since a performance measure is an estimate on a sample, how accurate is our estimate?
4. How to compare performances of different hypotheses or those of different classifiers?

Before we try to answer the issue 2 and 3

WE NEED TO UNDERSTAND WHAT ARE THE CAUSES OF AN ERROR (IN CLASSIFIERS AND REGRESSORS)

Why errors, in the first place?

- The task of a ML learning algorithm is to find an hypothesis model h(**x**) such that it approximates at best the real function *y=f(x)* both on the points **x1..xn** of our dataset D, **and on all other unseen examples**
- So h(**x**) must GENERALIZE on unseen examples
- However, perfectly fitting f(**x**) is impossible in most cases, as we said
- The errors (the difference between the real and learned functions) **is made up of 3 different components, as we have already seen:** bias, variance and irreducible error

Remember the 3 error components

Error=bias^2+variance+ irreducible error

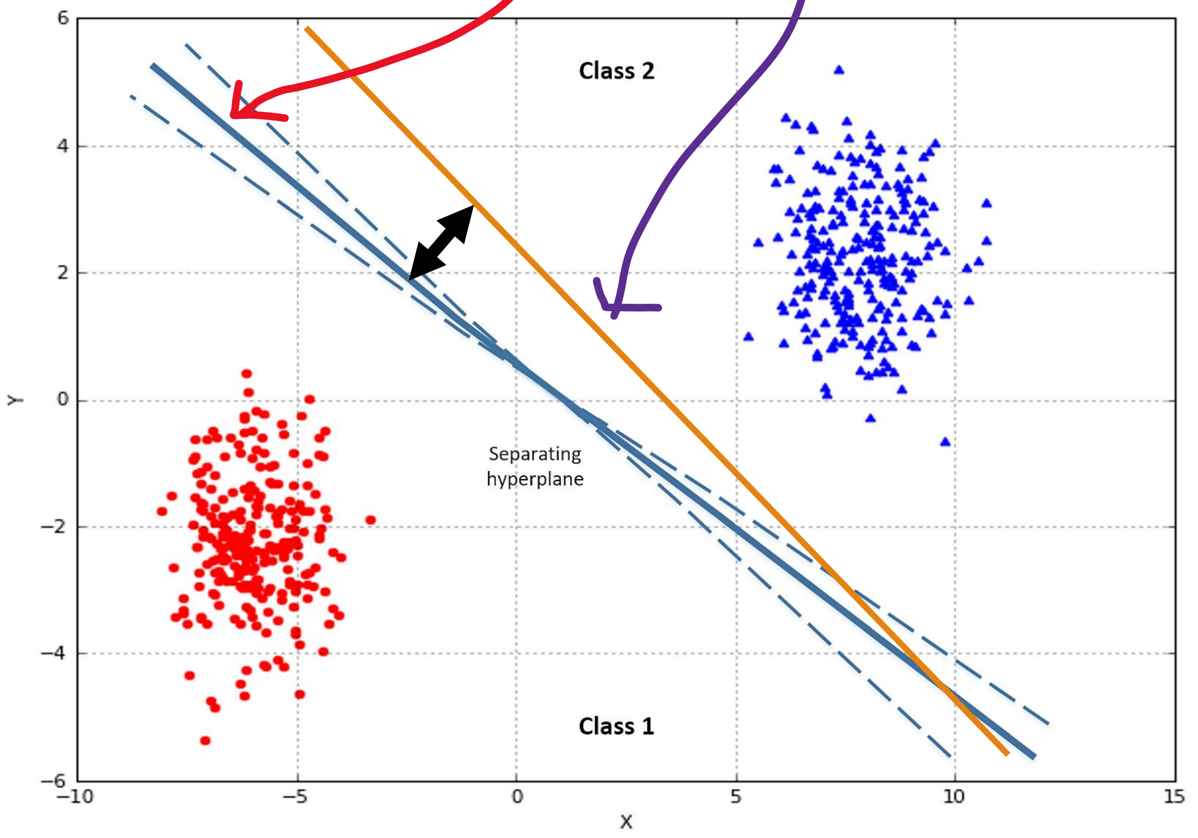
- The Bias lies in the algorithm (a tendency to model the problem in a specific way which might be inappropriate, e.g. a linear model for lineraly unseparable data). It can be expressed as:

$$Bias^2 = (E[h(x)] - f(x))^2$$

- Variance is the sensitivity of the model to the variability of the data: this can be reduced not only with ensambles, but also by using "appropriate" evaluation methods. It can be expressed as:

$$E[(h(x) - E(h(x))^2]$$

$$Bias^2 = (E[h(x)] - f(x))^2$$

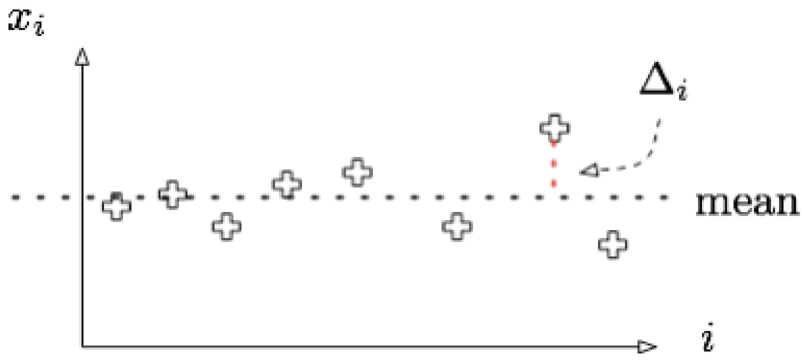


E[h(x)] is the expected value (mean) of different hypotheses obtained with different settings of the same model. Since it averages over different model settings, it is only sensible to the model choice

Variance
(for any discrete distribution)

Variance is defined (in general) as the mean of squared **differences** between values of **N** individual outcomes $\mathbf{x_i}$ and the mean (**x**), i.e. it measures the dispersion around the mean

$$\sigma^2 = var(x) = \frac{\sum_{i=1}^{N} (x_i - \bar{x})^2}{N}$$



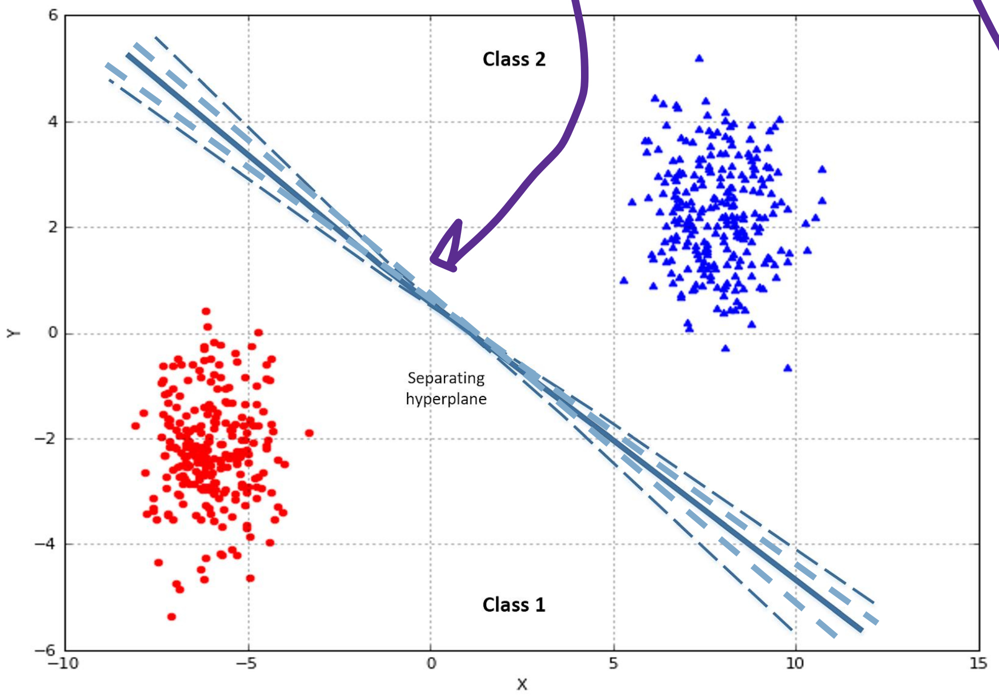
For continuous distributions, the sum becomes an integral

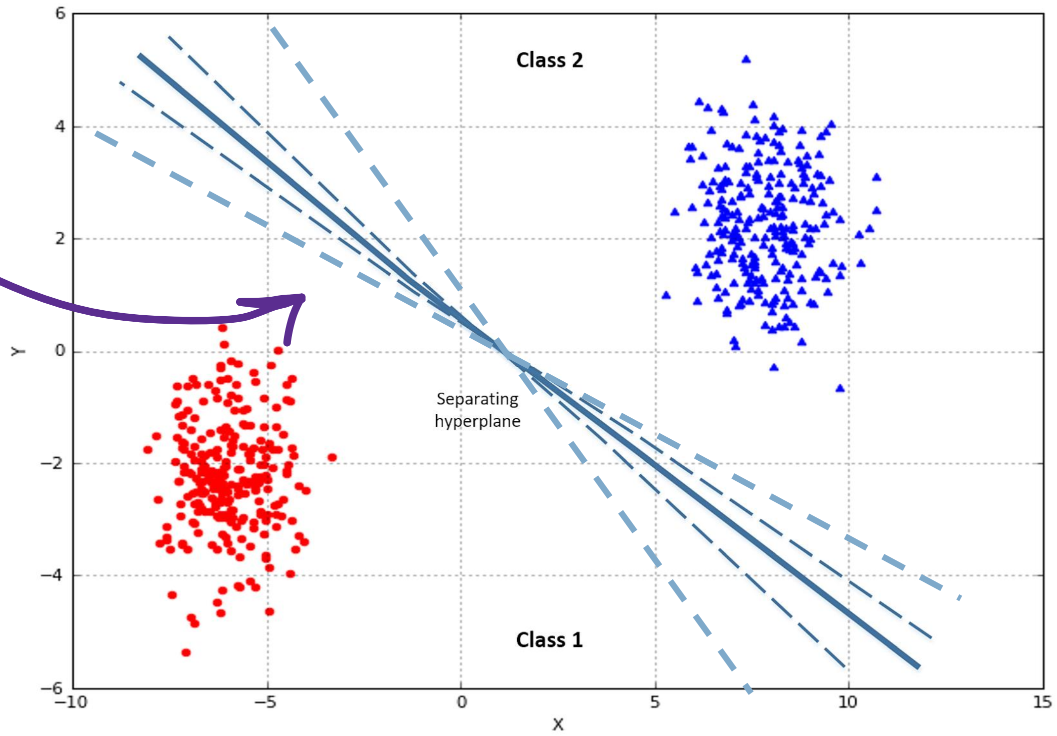
In our case, the variance is the «dispersion» of the **predicted output values** of the model *h(x)* around the mean

$$E[(h(x) - E(h(x))^2] = \sum_{i=1}^{n} (h(x_i) - E(h(x))^2$$

$$E[(h(x) - E(h(x))^2]$$

Lower variance Higher variance models





How to reduce the variance?

Variance cannot be reduced if inherent of our data.

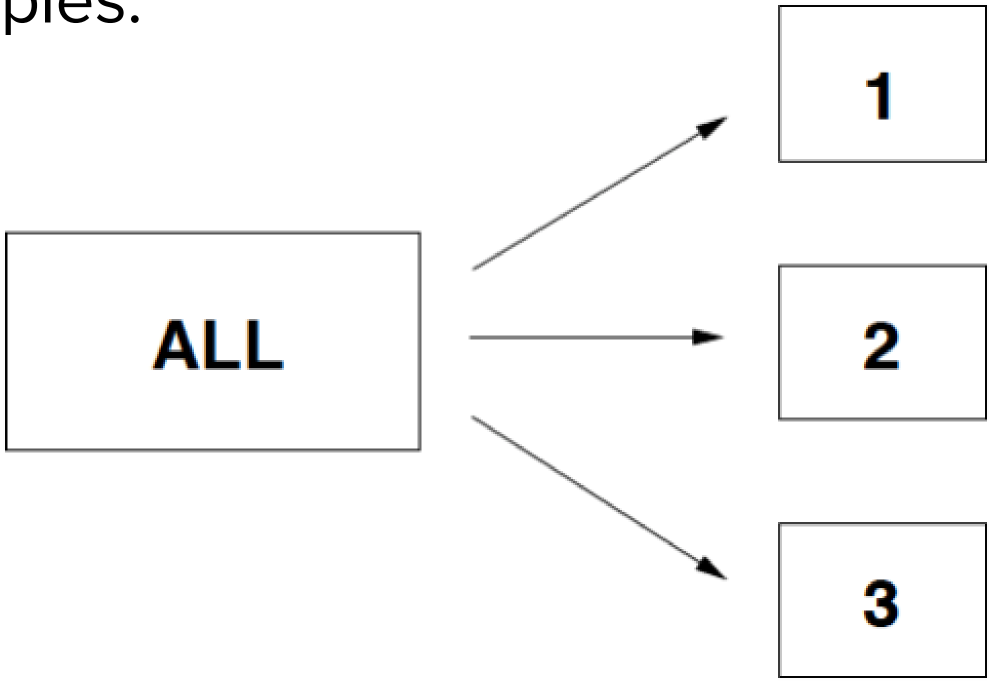
We can adopt two techniques (or a combination of the two):

- Train different models (each takes care of specific features of the data, we already presented this technique with **ensamble** methods)
- Use a **validation technique** that reduces the possibility of «being unlucky» when randomly selecting a test set: **K-Fold Cross-Validation:**

➢ K-FCV: perform several independent splits on learning and test set and then average the performance over these different splits.

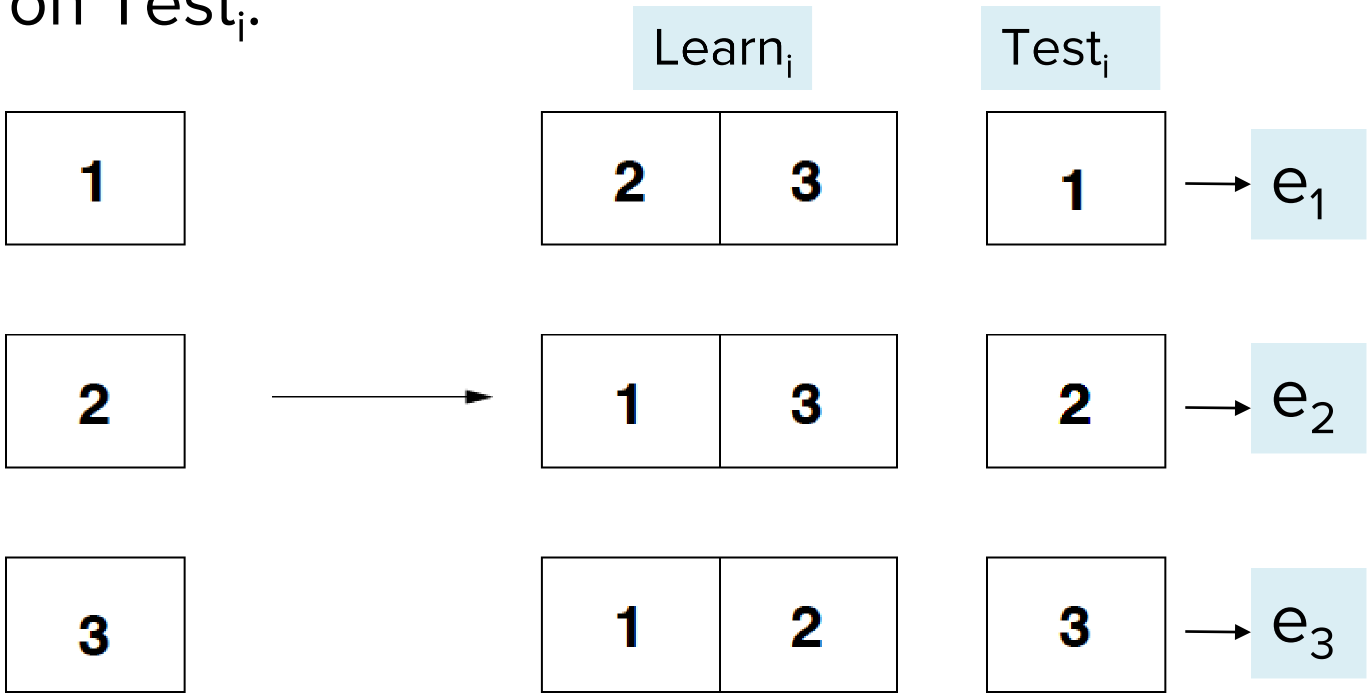
K-Fold Cross-Validation of a hypothesis (model) 1

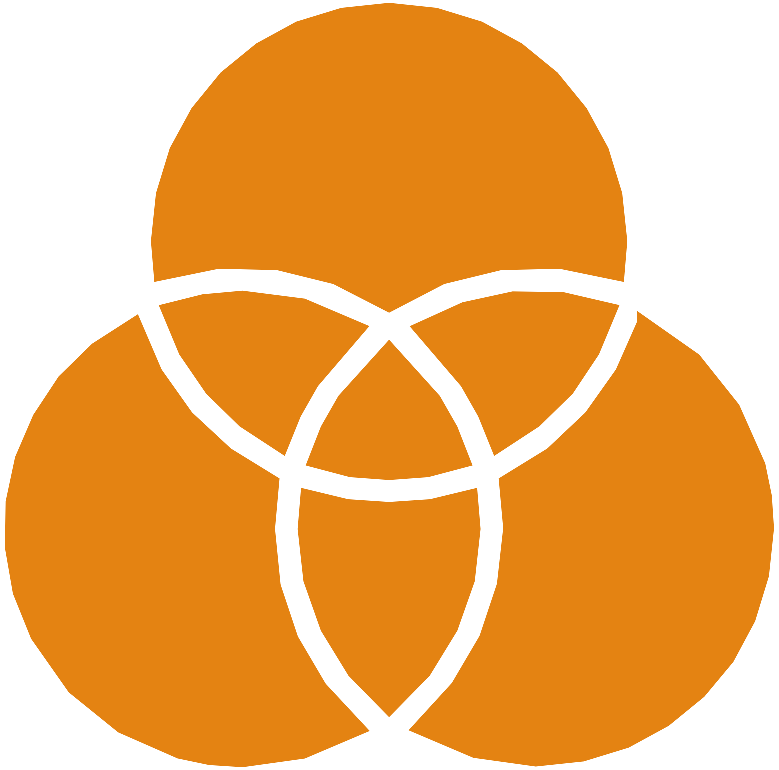
Partition all the available labeled data in **k equally sized** random samples.



K-Fold Cross-Validation of a hypothesis (model) 2

At each step i, learn from $Learn_i$ and test on $Test_i$, then compute the error (e_i) on $Test_i$.





Why K-Fold Cross Validation reduces the variance?

- Intuitively, it reduces the probability of "being lucky", or unlucky, in selecting the test-set
- To understand the issue more in detail, we need to introduce the next topic:
  - ➢ **testing the accuracy of an error estimate**

Variance and bias affect the results of performance measures

In practice, this means that, depending on the model, its sensitivity to changes in hyperparameters and to the choice of the training data, the performance measures can significantly vary.

So the question is: given this sensitivity, to what extent can we rely on performance evaluation experiments?

….Which brings us back to the initial questions:

1. Which performance measure we should use?
2. How well can a classifier be expected to perform on "novel" data, not used for training?
3. Since a performance measure is an estimate on a sample, how accurate is our estimate?
4. How to compare performances of different hypotheses or those of different classifiers?

Evaluation: What is an Estimator?

An Estimator is any function on a sample of the data that is used to *estimate* some «useful qualities» of the original data from which the sample is drawn. Formally, an *estimator* is a function on a sample ***S***:

$$\hat{\theta}_S = g(S), S = (x(1), \ldots, x(m)),$$

where ***x(i)*** is a random variable drawn from a distribution $\mathscr{D}$, i.e. ***x(i)*** ~ $\mathscr{D}$.

- We would like to use the sample S to ***estimate*** some useful qualities of the original data.
- For example, the *mean* is an estimator (mean value of a random variable X, given a sample of «trials»)
- In general, an estimator is any random variable used to estimate some parameter of the underlying population from which the sample is drawn
- An obvious question to ask about any estimator (not only the estimator of a ML error rate) is whether «on average» it gives the right estimate

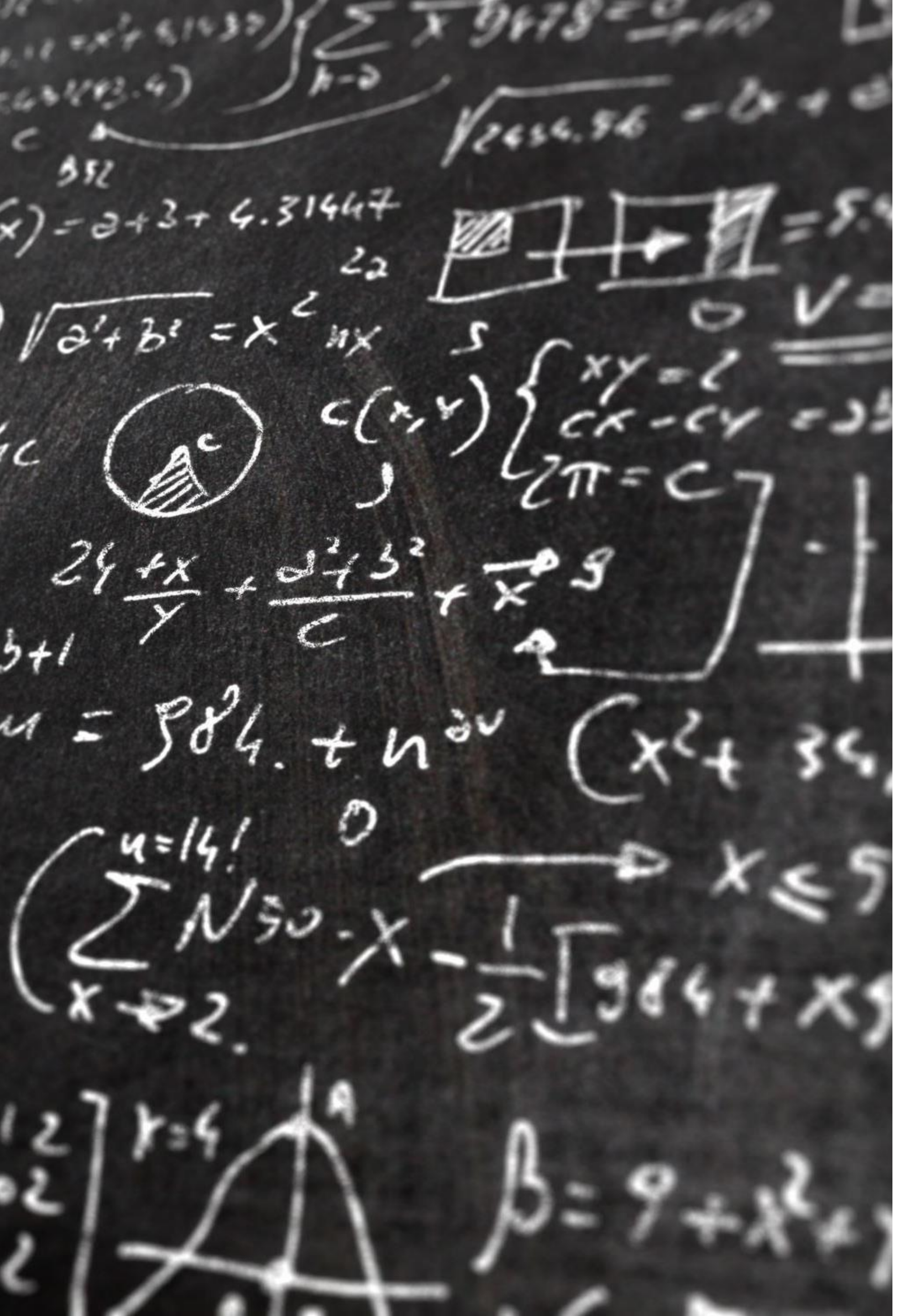
Questions to be considered in estimating the error of a model

Let *h(x)* be a model learned by a specific ML algorithm *L* using some specific hyper-parameters and choice of the training set D. The objective is **to estimate its prediction accuracy**. The following are relevant questions:

Q1: Given the observed accuracy (or any other performance measure) of *h* over a limited sample of test data **S**, **how well does this value estimate** its accuracy over additional (unseen) instances?

Q2: Given that one hypothesis h_1 outperforms another, h_2,over some sample data S, **how probable is** it that this hypothesis is more accurate **in general** (= over the full instance space)?

Note: we analyse the problem for classifiers, extending to regressors is straighforward



Estimating Hypothesis Accuracy

A better formulation of Q1:

A) Given a hypothesis *h* and a data sample containing *n* **instances** drawn at random according to distribution $\mathcal{D}$, **what is the best estimate** of the **accuracy** of *h* over **future instances** drawn from the same distribution?

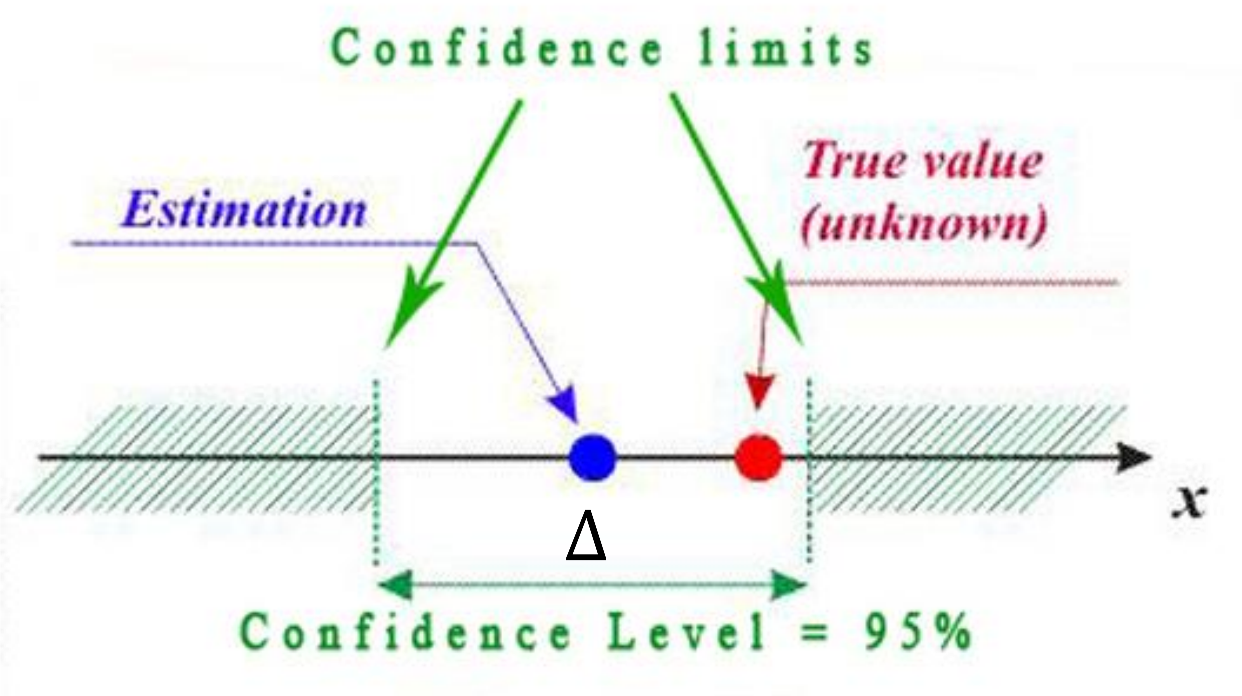
Need to consider: ***sample error*** vs. ***true error***

B) What is the «probable error» in this accuracy estimate?

Need to consider : ***confidence intervals*** (ranges in which the «true value» of the error may lie)

In other terms, if we measure an error rate (on a sample S) of, say, 20%, the **true error rate of *h* on any sample is not guaranteed to be exactly 20%**. Let's say that it is 20% ± Δ. Can we estimate the value of Δ (confidence interval)?

To answer our questions, we need to estimate the Confidence intervals



Objective: Estimating the interval around the estimated error, such that the true (unknown) error lies within these bounds with some confidence. (See later)

Sample Error and True Error

- **Definition *Sample Error*** (i.e., *error*$_s$*(h),* **error rate**):
  The ***sample error*** of hypothesis *h(x)* for the target function *c(x) (*the ground-truth classification c(x) of instances x in S), on a data sample *S of n instances* is:

$$error(h(x)) = \sum_{i=1}^{n} \delta(c(\boldsymbol{x_i}), h(\boldsymbol{x_i})) = r/n$$

where:

- *n* is the number of instances in sample *S*
- *r* is the number of misclassified instances
- *h(x)* is the classification produced by our current model *h*
- $\delta(c(x) \neq h(x)) = 1$ if $c(x) \neq h(x)$, and 0 otherwise.

Sample Error and True Error (2)

- **Definition True Error** (i.e., *error*$_{\mathcal{D}}$ *(h), p*):

  The ***true error*** of hypothesis *h* for the target (unknown) classification function *c(x)* and distribution $\mathcal{D}$ *of instances*, is the **probability** that *h* will misclassify **any** instance *x* drawn at random according to $\mathcal{D}$

$$\text{error}_{\mathcal{D}}(h)=\Pr(h(x)\neq c(x))$$

error$_s$*(h)* is an **estimator** of *error*$_D$*(h)*, which is a **probability**

So, how good is this estimator?

Remember, we consider classifiers c(x) for now, but it easilty extends to regressors f(x)

Estimate, probability and random variables

We are given a sample S of *n* instances, we classify S with *h(x)* and we measure *r* errors, we then estimate the error probability of *h(x)*:

$$\text{error}_S(h) = \hat{P}\,(r \text{ errors in } n \text{ instances}) = \frac{r}{n} = \mathbf{1 - accuracy}_{\boldsymbol{S}}\boldsymbol{(h)}$$

- *Note: We call S "sample" since it can be **any** subset X' of the set of instances X sampled according to a distribution $\mathcal{D}$.*

- However, *r (or* $\frac{r}{n}$ *)* is a **random variable**, governed by **chance**. If we choose another sample S' of *n* **different** instances, we may get a different number *r'* and a different estimate. In general $\mathbf{error_S(h) \neq error_{S'}(h)}$

A **Random Variable** can be viewed as the name of an experiment with a probabilistic outcome. Its value is the outcome of the experiment.

Estimate, probability and random variables

- **A simple experiment for a Random Variable**:
  - Make *k* different sets of trials, in each trial, toss a coin 10 times and measure the number of "head". Although, as the number of experiments *k* increases, the average number of "head" occurrences tend to k/2, in **every single trial** you will likely obtain different numbers.

- In coin tossing, we know that the "real" head rate (the expected value for the fraction of head tosses) is 50%, but in hypothesis testing, **we don't know what is the real error rate**. So, how can we get an idea of $error_{\mathcal{D}}(h)$ on the entire population X, distributed according to $\mathcal{D}$?

Sample Error & True Error (3)

Our question is: Is the Sample Error a good estimator for the True Error?

We do not know the “true” error probability however we know that **$\mathbf{error_s(h)}$** is a **random variable** that follows a **binomial distribution** with mean ***p*** (the unknown «true» expected error)

What is this “binomial”?

Sample Error & True Error:

Why a binomial?

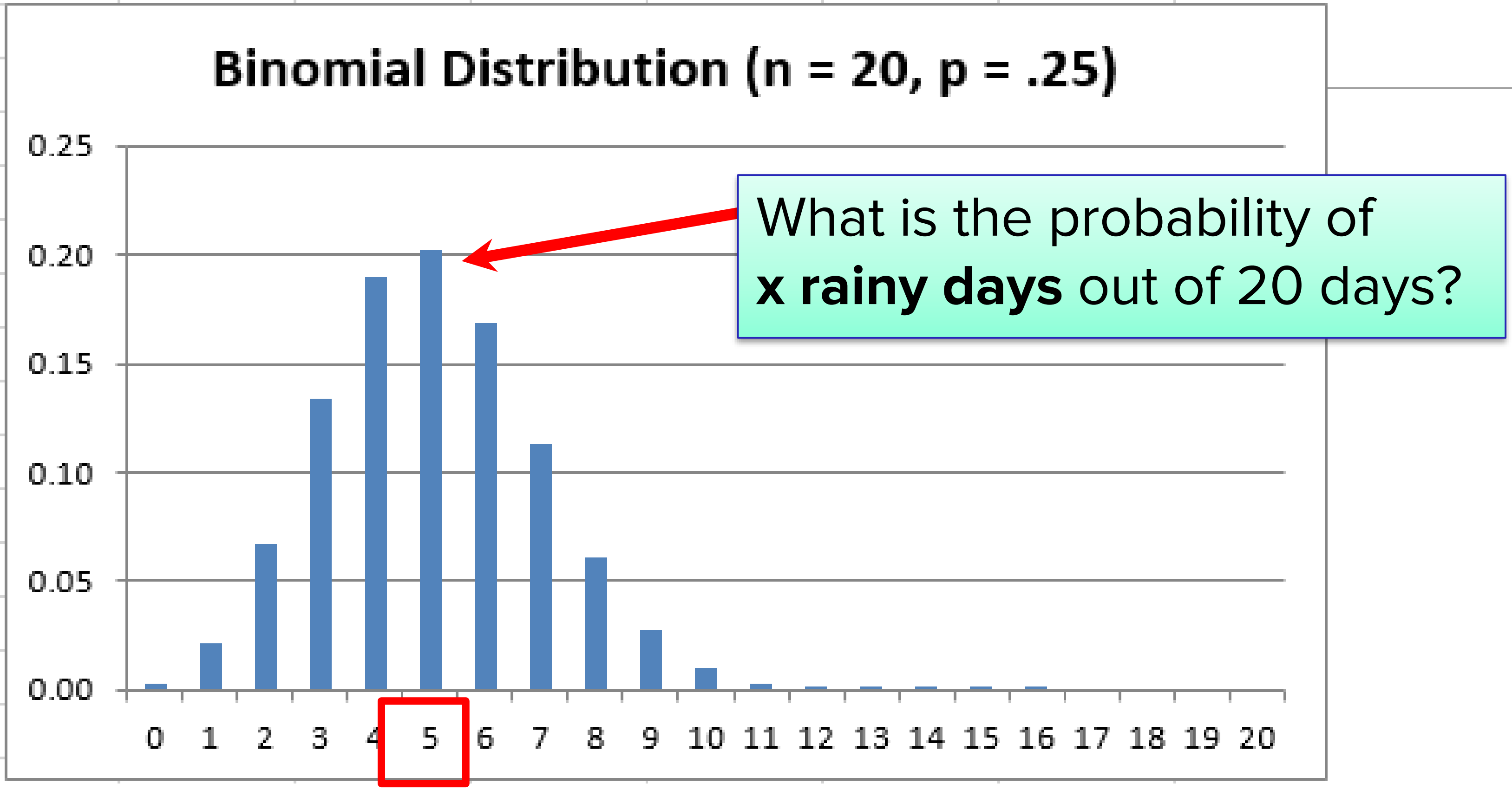
- Say p is the (unknown) "true" expected error of h(x) on X. If we have a sample S of n instances (test set), what is the probability that, given instances x in S, $c(x) \neq h(x)$ for r times??
- Even if we do not know the true value of p *(the expected value of the error)*, each instance x in S has probability ***p*** of being misclassified by h(x) and ***(1-p)*** of being classified correctly.
- The probability of observing r misclassified examples in n instances is then:

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

# of ways in which we can select **r** items from a population of **n**

Example:

p is the probability of rain days in January



The abscissa is the value **r** (n. of rainy days), on the y axis we read the correspondent probability, e.g. there is a 20% probability that there will be 5 **rainy days** out of 20 observations, 6% probability of 8 rainy days out of 20, etc.

How do we compute these probabilities?

- Say we know that *p(rain)* = 25% (on January)

- However, if we watch the weather in 4 consecutive days, **we are not sure** we will get “rain” 1 time and “not rain” 3 times. The number of observed “rainy days” in each trial of 4 consecutive days **is governed by chance**.

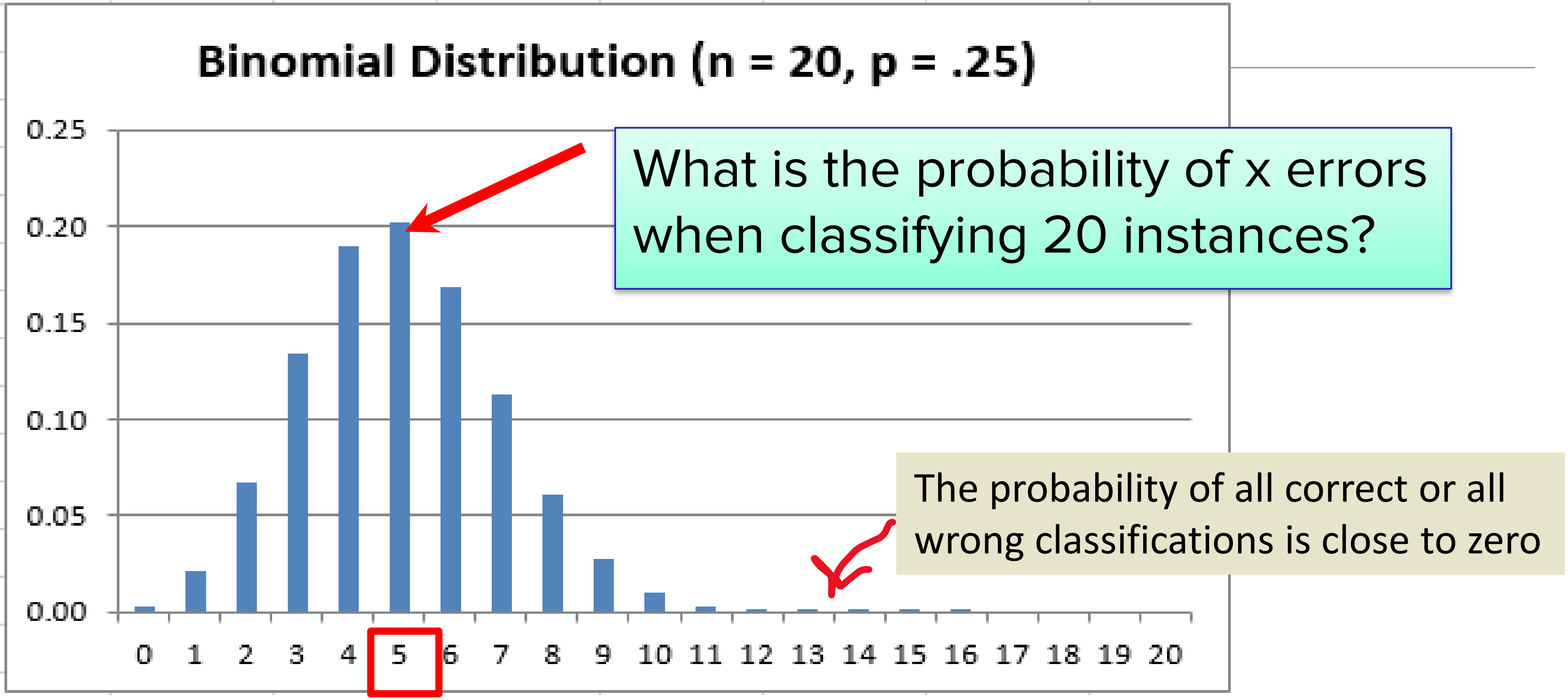
- What is the probability of getting, instead, 2 rainy days in 4 days?

$$P(2\ \text{"rain"}\ in\ 4\ observed\ days) = \binom{4}{2}(0.25)^2(1-0{,}25)^2 =$$

$$= \frac{4!}{2!\,(4-2)!}(0.25)^2(1-0{,}25)^2 = \frac{1\cdot 2\cdot 3\cdot 4}{1\cdot 2(1\cdot 2)}0.065(0.5625) = 0{,}21$$

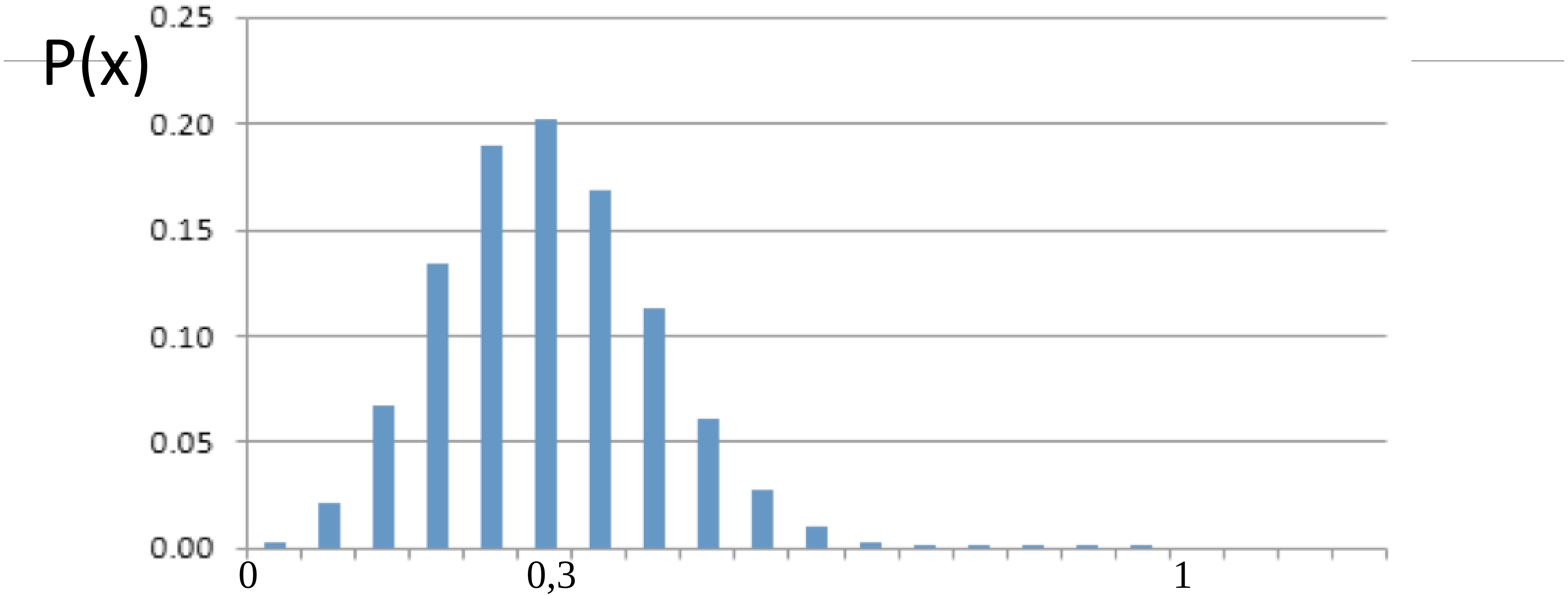
Same formula to estimate the probability of 2 **errors** over 4 instances, given we know that the true error rate is 25%

Example: ***p is the expected value of the probability that our ML system misclassifies an instance x drawn at random from the entire population of instances***



- The abscissa is the value **r**, e.g. there is a 20% probability that there will be 5 **errors** out of 20 classifications, 6% probability of 8 out of 20, etc.

We usually normalize and plot r/n



Now x is the % of wrongly classified instances

Even if we do not know p, we know that if we perform several experiments on different samples S (test sets) we will observe a **bell shape** distribution of the error rate r/n!!!!

Properties of Binomial distribution

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

Probability of r errors in n trials

- Expected Value of r over n trials: $E(X) = np$
- Variance: $Var(X) = np(1-p)$
- Standard Deviation (STD, SD): $\sigma(X) = \sqrt{np(1-p)}$

o_i is the outcome of a correctness test on instance xi. It is 1 If c(x)≠ h(x) and 0 if c(x)=h(x)

$$Var\,(X) = \frac{\sum_{i=1}^{n}(o_i - p)^2}{n} = \frac{1}{n}\left(np(1-p)^2 + n(1-p)(0-p)^2\right)$$
$$= np(1-p)$$

For *np* times o=1, for *n(1-p)* times o=0

Estimator of an error

Now, we know that the random variable $X=r$ (number of errors observed in n independent tests) follows a **binomial distribution** with unknown mean **np**. If we compute the error *rate* on a sample of n observations S, we obtain a **value** r/n which is our current **estimate** $error_S(h)$ of $error_{\mathcal{D}}(h)$.

- **Note** that the "estimator" $error_S(h)$ is also a **random variable**! If we perform many experiments on **different samples** Si we could get different values.
- However, **for large enough dimension of the sample S**, the expected value of $error_S(h)$ (i.e. $E[error_S(h)]$) **is the same** as for $error_{\mathcal{D}}(h)$!

Why? Because of the ***Central Limit Theorem***

Central Limit Theorem

General Formulation:
The theorem states that the arithmetic mean of a sufficiently large number of experiments of independent random variables, each with a well-defined expected value and well-defined variance, will be approximately normally distributed.

- This will hold **regardless of whether the source population is normal or skewed (biased)**, provided the samples size is sufficiently large (usually $n \geq 30$ but this is a "rule of the thumb" , and a better way of establishing a threshold can be found here)
- Furthermore, **the mean of all such experiments will (tend to) be the same as the "real" population mean**
- **A Normal distribution (or Gaussian Distribution)**:

  A family of continuous probability distributions such that the probability density function is the normal (or Gaussian) function

Putting it all toghether:

$error_S(h)$ and $error_{\mathcal{D}}(h)$ both follow a gaussian law, and $E(error_S(h)) \rightarrow p$

In our case we know:

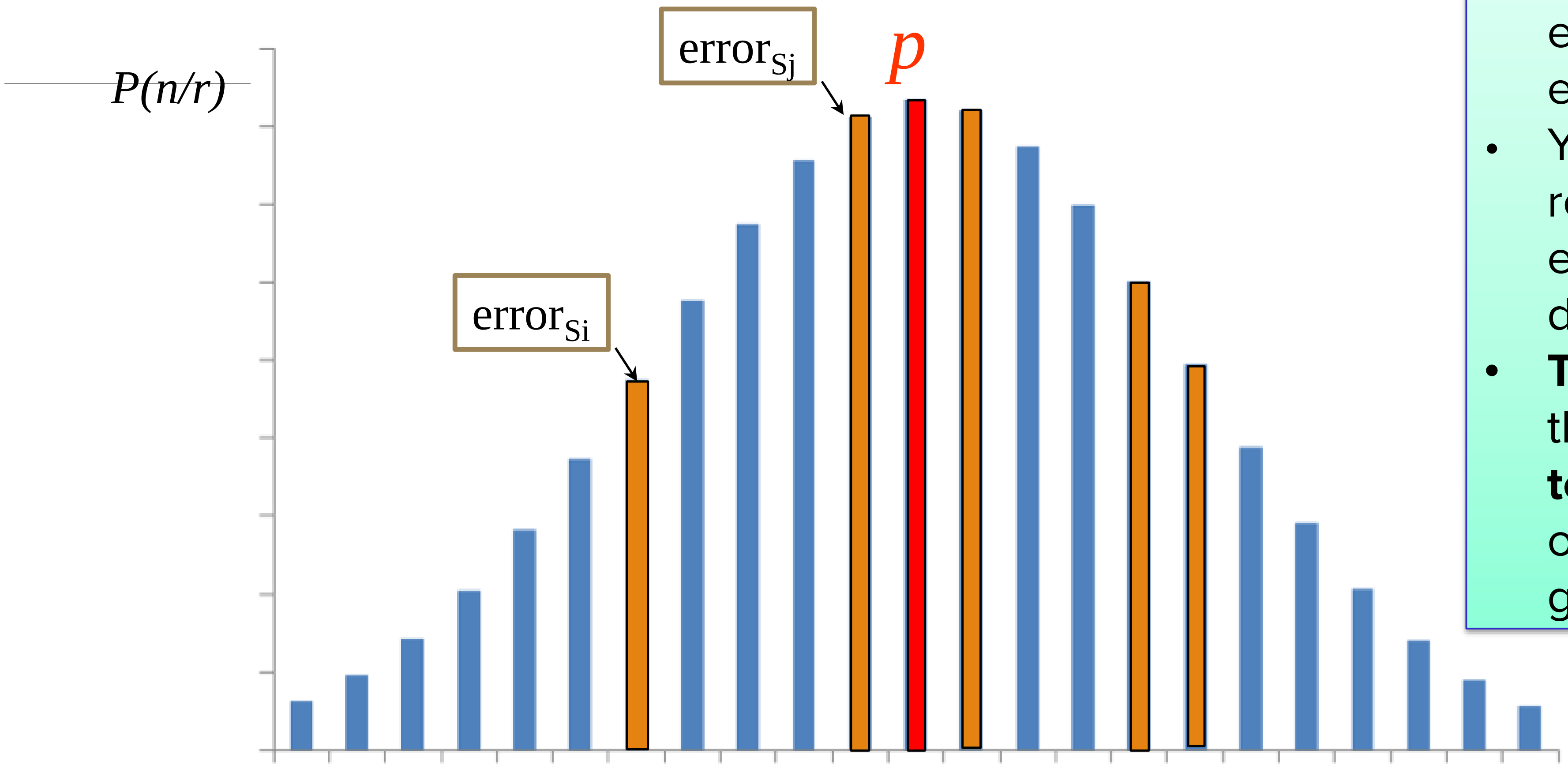
a) **Experiments are accuracy tests** on data samples S_i

b) The involved **random variables are the error rates** r_i/n_i observed on these samples S_i. These random variables are statistically **independent** of each other, and **follow a binomial** distribution, as we have seen

c) For a **sufficiently large number of experiments**, the observed values r_i/n_i will be approximately **normally distributed**, according to the central limit theorem

d) Their **mean value** will tend to the "real" (the unknown true value) expected error **p** over the entire set of instances X

mean(error$_S$(h))➔p=E[error$_{\mathcal{D}}$(h(x))]



- *p* is the unknown expected (true error) error rate.
- Yellow bars are the results of different experiments on different samples S_i
- **The average** of these results **tends to p** as the number of **experiments** grows

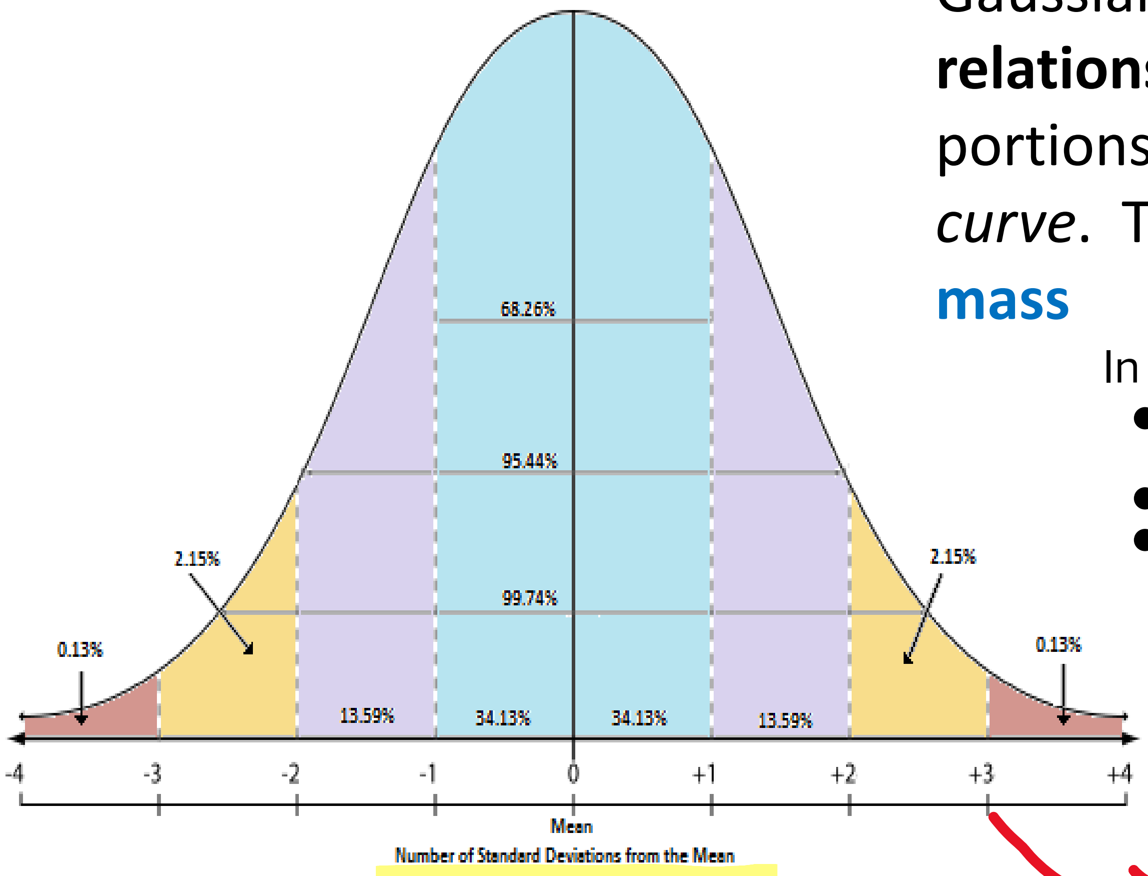
The average of many observed values of the random variable error$_S$(h), generated by repeated random experiments, converges toward p, the expected value of the TRUE arror rate over the entire distriibution $\mathcal{D}$ of instances.

Gaussian (normal) Distribution

The curve parameters, as we have already seen, are the mean **μ** (e.g., **p** - the expected error rate - in our specific case) and the standard deviation **σ**.

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

Interesting properties of gaussian distributions



Gaussian curves allow to establish a **fixed relationship** between standard deviation and portions of the *area under* the Gaussian *curve*. These areas are interpreted as **probability mass**

In a gaussian curve, **for any μ and σ,** it holds that:

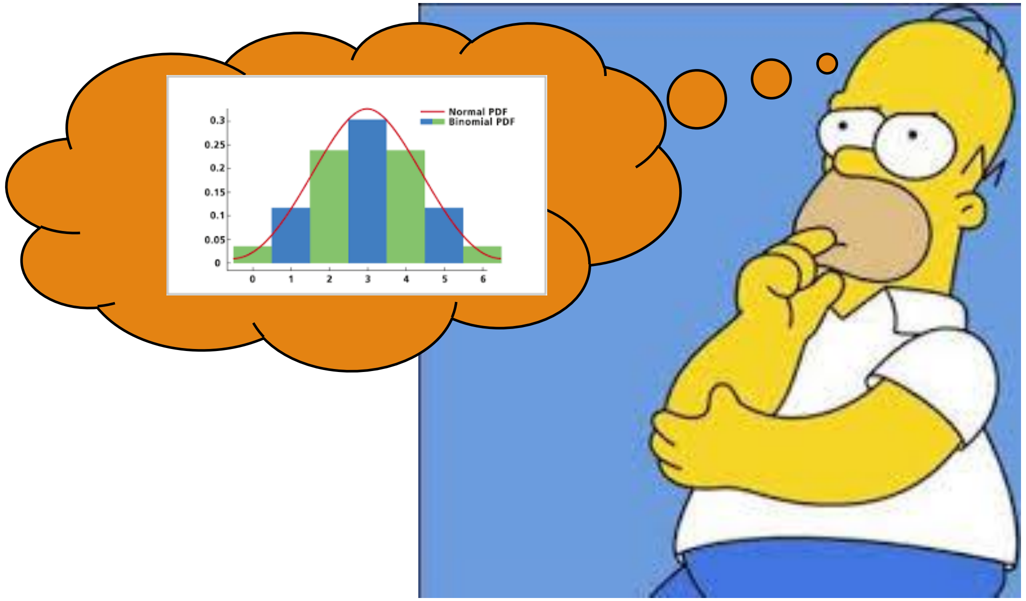
- **99.7%** of the **probability mass** lies in the area below the mean value **μ ± 3σ**
- **95.4%** of the probability mass lies in the area below **μ ± 2σ**
- **68.3%** of the probability mass lies in the area below **μ ± σ**

3xσ

Number of standard deviations from the mean (z value)

but..when shall we get to our problem of estimating the quality of an error estimator ???

Stay tuned pls..



Consequences of applicability central limit theorem to the random variable r/n

Result 1:

If the random variable X=r/n follows a Gaussian distribution, then $error_S(h(x))$=r/n **is an unbiased estimator** of the real expected error rate p since:

Bias is defined as the systematic deviation of a quantity from the actual value.

In our case, we are talking about the bias of the error function:

$$(Bias(error_s(h))^2 = (E[error_S(h)] - error_D(h))^2 = (p-p)^2 = 0$$

Consequences of applicability central limit theorem to the random variable r/n

Result 2: we can approximate the standard deviation of error$_{\mathcal{D}}$(h(x))

The Standard deviation **of a sample S of n** instances is defined as:

$$\sigma_S = \frac{\sigma_r}{n} = \frac{1}{n}\sqrt{np\cdot(1-p)} = \boxed{\sqrt{\frac{p\cdot(1-p)}{n}} \simeq \sqrt{\frac{\frac{r}{n}\cdot(1-\frac{r}{n})}{n}}} = \sqrt{\frac{error_s(h)\cdot(1-error_s(h))}{n}}$$

- **Note** that for $n \to \infty$ (very large samples), then $\boldsymbol{\sigma_S \to 0}$

(since $r/n \to p$ i.e., the **observed error will converge to the real error rate**)

We replace the (unknown) p with our computed mean value r/n. This is an **estimate** since we assume that r/n is a good approximation of the real error rate p, which holds approximately true **for large enough n**, according to CLT!

Why is this approximation acceptable (and replacing p with r/n is not)?

- Why we can set $p(1-p) \simeq \frac{r}{n}(1-\frac{r}{n})$??
- Say p=0.6 and r/n=0.7 (difference is 0.1)
- However, p(1-p)=0.24 r/n(1-r/n)=0.21 (difference is only 0.03))
- → Although approximating the real error with the estimated error can lead to a significant over or under-estimate, **approximating the real SD with the estimated SD is much less critical**
- In general, if **n is sufficiently large**, the probability that our estimate is very far from real SD is sufficiently low

Consequences of applicability central limit theorem to the random variable r/n

Result 3:

Normal (gaussian) distributions have **important properties** concerning how the probability mass is distributed below the curve **(e.g., 99.7%** of the **probability mass** lies in the area below the mean value **$\mu \pm 3\sigma$, 95.4%** of the probability mass lies in the area below **$\mu \pm 2\sigma$, 68.3%** of the probability mass lies in the area below **$\mu \pm \sigma$ …)**, establishing fixed relationships between the probability mass and intervals around the mean.

This property allows easy calculation of confidence intervals!!

We are ready to compute the confidence intervals for an error estimate

Confidence interval for an error estimate

- The confidence interval represents the statistical significance (**Margin Error**; ME) of the expected **distance *Δ*** between the real value (in our case, p) and the observed estimate (in our case, r/n).
- **Definition**: An N% confidence interval for some parameter ***p*** **is an interval** [LB, UB] that is expected with probability N% to contain p. (equivalently: *with probability N% we have LB≤* p *≤UB*)
- The confidence interval is a way to show what the uncertainty is with a certain measured statistics. The margin of error ME tells you **how many percentages points your results** (e.g., your estimated error rate) **will differ** from the real population value (e.g., the real error rate)

Confidence interval for an error estimate

- **Confidence interval (CI)**

$$\Delta = |\, error_D(h) - error_S(h)| \leq ME \Rightarrow |\, p - \tfrac{r}{n}| \leq ME$$

$$\left|p - \frac{r}{n}\right| \leq z\sigma \simeq z\sigma_S \Longrightarrow \frac{r}{n} - z\sigma_S \leq p \leq \frac{r}{n} + z\sigma_S$$

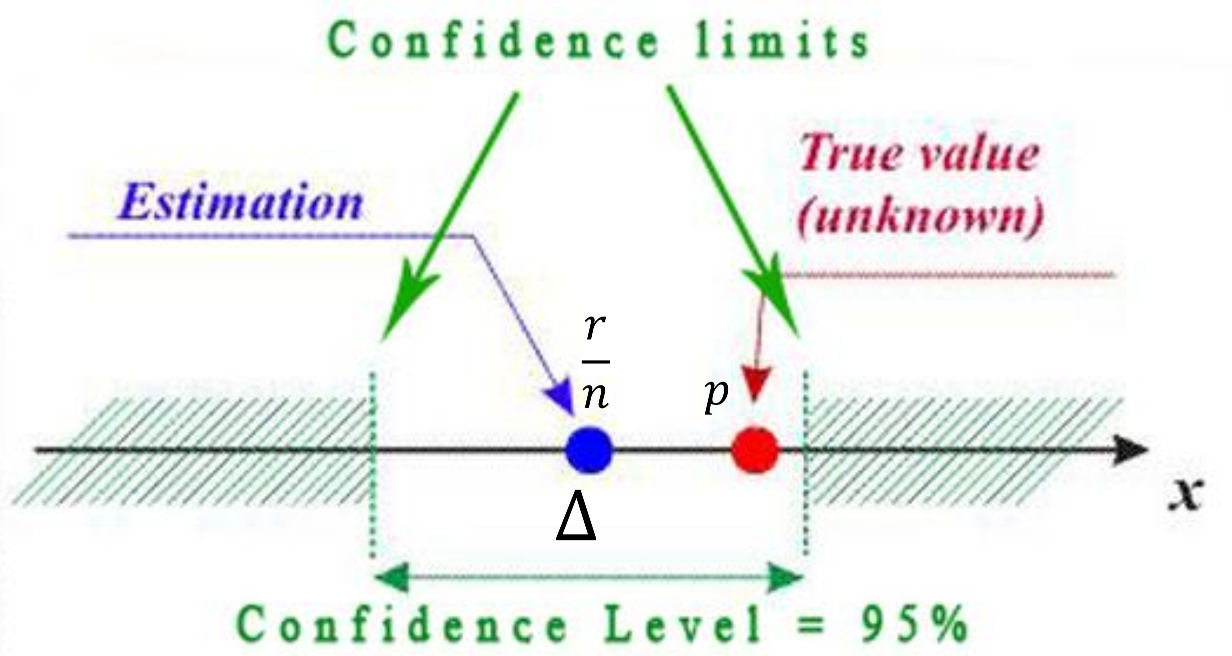
➢**ME (Margin Error)** = $\boldsymbol{z\sigma}$= (Critical Value) x (Standard Deviation for the population)

The critical values are also called z-values.

➢***Δ*** is called **Absolute Error** of the estimate

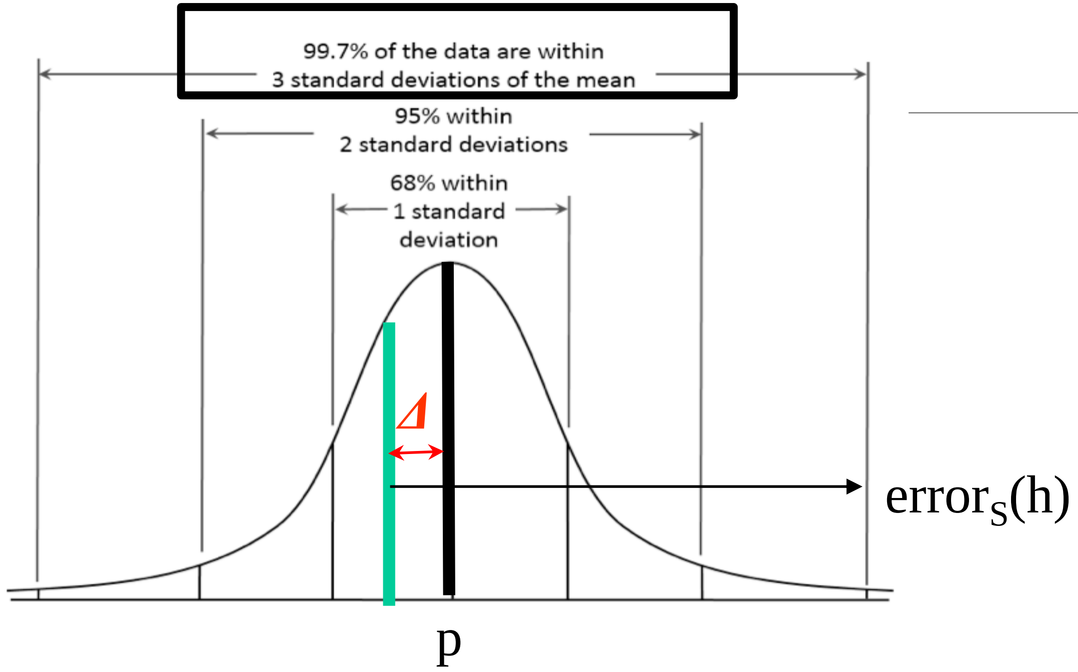
We are expressing the interval in terms of «how many» standard deviations. z is the (unknown) «how many»

Confidence intervals



The good news is that the error follows a gaussian distribution, a regular and symmetric distribution that facilitates the computation of such intervals

Confidence intervals computation with Gaussian Distributions



$\boldsymbol{\Delta}$ = | $error_D(h(x))$ - $error_S(h(x))$|

For any gaussian, we can say "with a probability of 68% (95%, 99.7%, **N%**) **any value x we measure for $error_S$** will lie in the interval ±1σ (±2σ, ±3σ, **±zσ)** around the mean **p**". More in general, **with an N% probability it will lie in the ±zσ interval**

One reason that we prefer to work with the gaussian distribution is that we have tables specifying **the size of the interval around the mean that contains N% of the probability mass** under the Normal distribution. This is precisely the information (**the critical values, z-values**) needed to calculate our N% confidence interval.

Since $error_S(h(x))$ follows a gaussian, we can use this property!

How do we compute confidence intervals in practice?

- We must fix either N% (the confidence, or probability mass) or z (the lenght of the interval, in terms of «how many» standard deviations, or "critical value")
- Clearly, the higher is the confidence we need, the larger is the interval we will find
- We must set as our target **either** the confidence, **or** the lenght of the interval
- For gaussian curves, tables are provided to determine one variable when the other is given, e.g. :

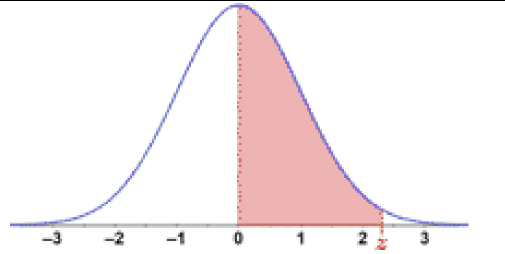
N%	50	68	80	90	95	98	99
z_N	0,67	1.00	1.28	1.64	1.96	2.33	2.58

- Tables are provided to compute z for any N and viceversa
- To compute confidence intervals from z tables, see here

e.g. with 90% probability, The true error will lie in an interval of +/- 1.64σ around the estimated error rate

How to use z-tables to relate N and z

Z-table



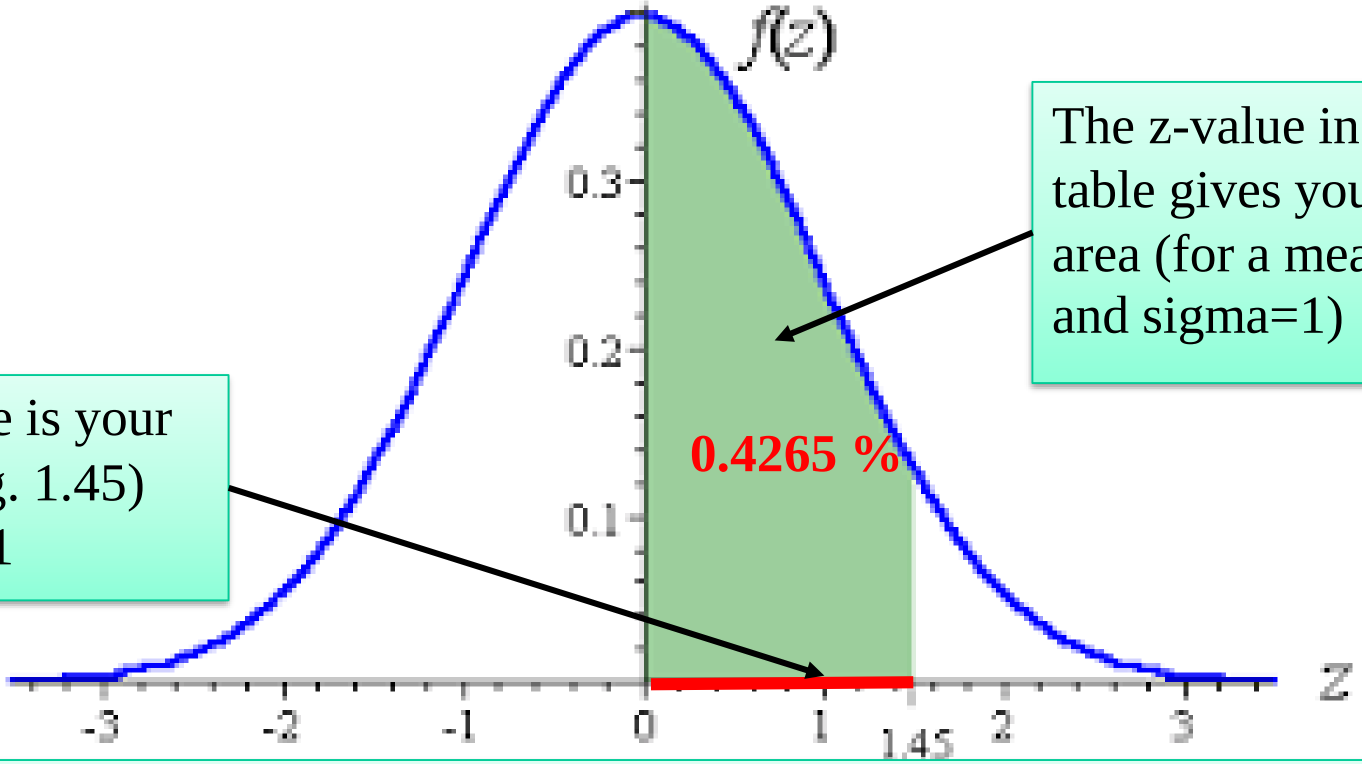
Entries in the table give the percentage of the area under the curve from zero to the z-score. For example, if $z = 1.32$ the area under the curve between 0 and 1.32 is 0.40658.
If $z = -1.32$, the area under the curve between −1.32 and 0 is still 0.40658.

±z	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0	0	0.00399	0.00798	0.01197	0.01595	0.01994	0.02392	0.0279	0.03188	0.03586
0.1	0.03983	0.0438	0.04776	0.05172	0.05567	0.05962	0.06356	0.06749	0.07142	0.07535
0.2	0.07926	0.08317	0.08706	0.09095	0.09483	0.09871	0.10257	0.10642	0.11026	0.11409
0.3	0.11791	0.12172	0.12552	0.1293	0.13307	0.13683	0.14058	0.14431	0.14803	0.15173
0.4	0.15542	0.1591	0.16276	0.1664	0.17003	0.17364	0.17724	0.18082	0.18439	0.18793
0.5	0.19146	0.19497	0.19847	0.20194	0.2054	0.20884	0.21226	0.21566	0.21904	0.2224
0.6	0.22575	0.22907	0.23237	0.23565	0.23891	0.24215	0.24537	0.24857	0.25175	0.2549
[illegible]	.25804	0.26115	0.26424	0.2673	0.27035	0.27337	0.27637	0.27935	0.2823	0.28524
[illegible]	.28814	0.29103	0.29389	0.29673	0.29955	0.30234	0.30511	0.30785	0.31057	0.31327
0.9	0.31594	0.31859	0.32121	0.32381	0.32639	0.32894	0.33147	0.33398	0.33646	0.33891
1	0.34134	0.34375	0.34614	0.34849	0.35083	0.35314	0.35543	0.35769	0.35993	0.36214
1.1	0.36433	0.3665	0.36864	0.37076	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1.2	0.38493	0.38686	0.38877	[illegible]	0.39251	0.39435	0.39617	0.39796	0.39973	0.40147
1.3	0.4032	0.4049	0.40658	0.40824	0.40988	0.41149	0.41308	0.41466	0.41621	0.41774
1.4	0.41924	0.42073	0.4222	0.42364	0.42507	0.42647	0.42785	0.42922	0.43056	0.43189
1.5	0.43319	0.43448	0.43574	0.43699	0.43822	0.43943	0.44062	0.44179	0.44295	0.44408
1.6	0.4452	0.4463	0.44738	0.44845	0.4495	0.45053	0.45154	0.45254	0.45352	0.45449
1.7	0.45543	0.45637	0.45728	0.45818	0.45907	0.45994	0.4608	0.46164	0.46246	0.46327
1.8	0.46407	0.46485	0.46562	0.46638	0.46712	0.46784	0.46856	0.46926	0.46995	0.47062

$z = z_{row} + z_{column}$

N in the cell must be multiplicated by 2

The Z-table:
Gaussian Distribution



The z-value in the table gives you this area (for a mean =0 and sigma=1)

The red line is your z-value (e.g. 1.45) for sigma=1

Or viceversa if the input is N: Dividing by 2 the probability mass (say, N= 85.3% /2 = 0.4265), we obtain the z (1.45) value from the table, to calculate the interval

How to : Finding the N% confidence interval

- We know the formula to compute the interval, **given** the estimated error rate:

$$[LB, UB] = \left[\frac{r}{n} - z\sqrt{\frac{r/n(1-\frac{r}{n})}{n}}, \frac{r}{n} + z\sqrt{\frac{r/n(1-\frac{r}{n})}{n}}\right]$$

- In this formula, **z** is unknown. But we fixed N, so we look in the table and we obtain z for the desired N, and compute the interval.

Example 1

- We have a classifier which produced a hypothesis model h(x), and a test set S of 100 instances
- We apply h(x) on the sample test set S and compute 13% (0.13) error rate (r/n)
- Since n>30 **we assume that the error distribution follows a gaussian distribution** with mean 0,13 and standard deviation σ_S:

$$\sqrt{0.13(1-0.13)/100}$$

- To compute the N=90% confidence interval, on the table we find Z=1.64

N%	50	68	80	90	95	98	99
z_N	0,67	1.00	1.28	1.64	1.96	2.33	2.58

Example 1:
Calculating the N% Confidence Interval

- We then have:

  Z=1.64 and $\sigma_S \simeq \sqrt{0.13(1-0.13)/100}$

- The 90% confidence interval is estimated using the previous formula is:

$$\left[0.13-1.64\sqrt{\frac{0.13(1-0.13)}{100}}, 0.13+1.64\sqrt{\frac{0.13(1-0.13)}{100}}\right] = [0.075, 0.19]$$

Example 2:

Finding 95% CI on a face recognition task

Given the following extract from a scientific paper on multimodal emotion recognition:

We trained the classifiers with 156 samples and tested with 50 samples from three subjects.

⋮

Table 3. Emotion recognition results for 3 subjects using 156 training and 50 testing samples.

	Attributes	Number of Classes	Classifier	Correctly classified
Face*	67	8	C4.5	78 %
Body*	140	6	BayesNet	90 %

For the Face modality, what is n? What is $error_s(h)$?

N%	**50**	**68**	**80**	**90**	**95**	**98**	**99**
$\mathbf{z_N}$	**0,67**	**1.00**	**1.28**	**1.64**	**1.96**	**2.33**	**2.58**

Example 2:

Accuracy is 0.78, hence **error rate** is 0.22; the test set has 50 instances, hence **n=50.**

Choose, e.g., to compute the N% confidence interval with **N=0.95**

Given that $error_s(h)$=0.22 and n= 50, and z_N=1.96 for N = 95, we can now say that with 95% probability $error_D(h)$ will lie in the interval:

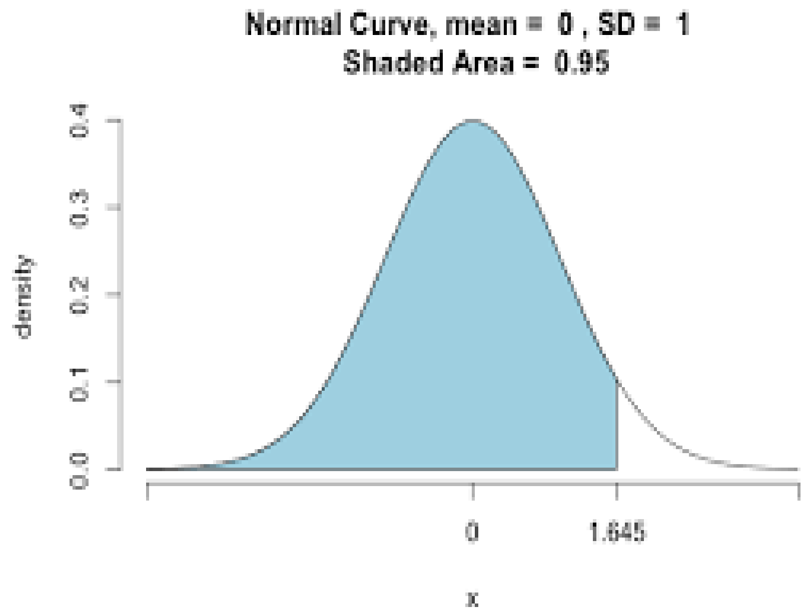
$$\left[0.22 - 1.96\sqrt{\frac{0.22(1-0.22)}{50}}, 0.22 + 1.96\sqrt{\frac{0.22(1-0.22)}{50}}\right] =$$

$$[0.11, 0.34]$$

From here on not on 23-24

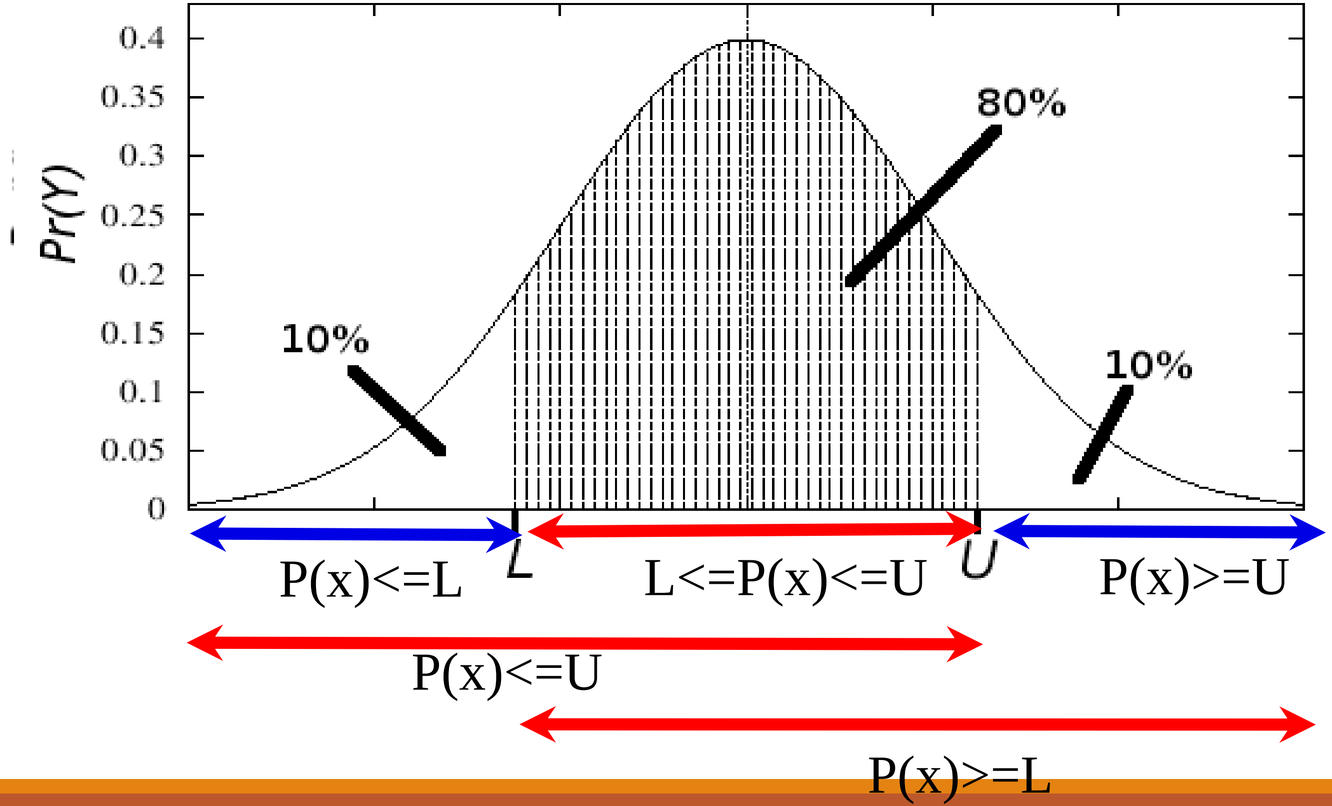
One side bound (Tailed-test)

- We might be interested in computing the probability that the **error** of our ML system **is "at most" a given value**, **rather** than **within a given range** like before.
- Which amounts to computing the blue area
- Now **N%** is the area for which $error_S$ **≤ zσ**



Gaussian is symmetric!

One-sided / Two-sided bounds: The Gaussian distribution is symmetric and its total area is 100% (of the probability mass).

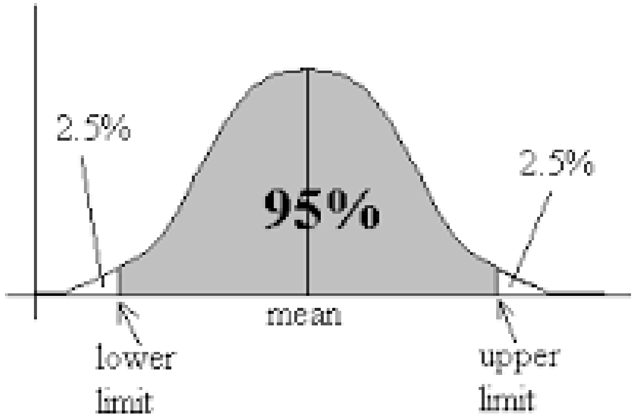


Example: One/Two-Sided bounds

In the previous emotion recognition example, we said that with 95% probability (the confidence), the true error (i.e $error_D$) lies in the [0.11,0.34] interval.

- There is a 5% **(100-95=5)** area outside this interval, of which, 2.5% to the left and 2.5% to the right (due to symmetry)
- Therefore, we can also say that there is a 2.5% probability that $error_D > 0.34$ (the upper bound UB) and 2.5% probability that $error_D < 0.11$ (the lower bound LB)

- There is a 97.5% (95+2.5=97.5) probability that: $error_{\mathcal{D}} < 0.34$
- There is a 97.5% (95+2.5=97.5) probability that: $error_{\mathcal{D}} > 0.11$



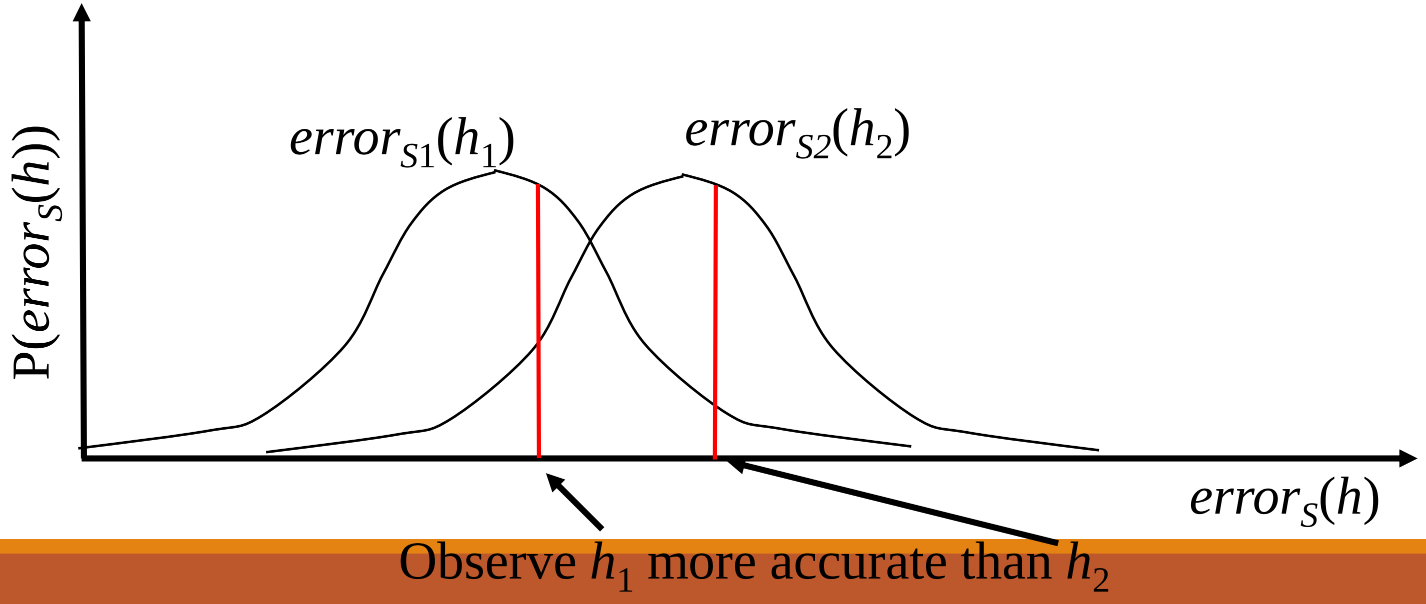
Issues

1. Which performance measure we should use?
2. How well can a classifier be expected to perform on “novel” data, not used for training?
3. Since a performance measure is an estimate on a sample, how accurate is our estimate?
4. **How to compare performances of different hypotheses or those of different classifiers?**

Comparing Two Learned Hypotheses

When evaluating two hypotheses (e.g. using different hyper-parameters on the **same ML algorithm**), their observed ordering concerning accuracy **may or may not** reflect the ordering of their **true** accuracies.

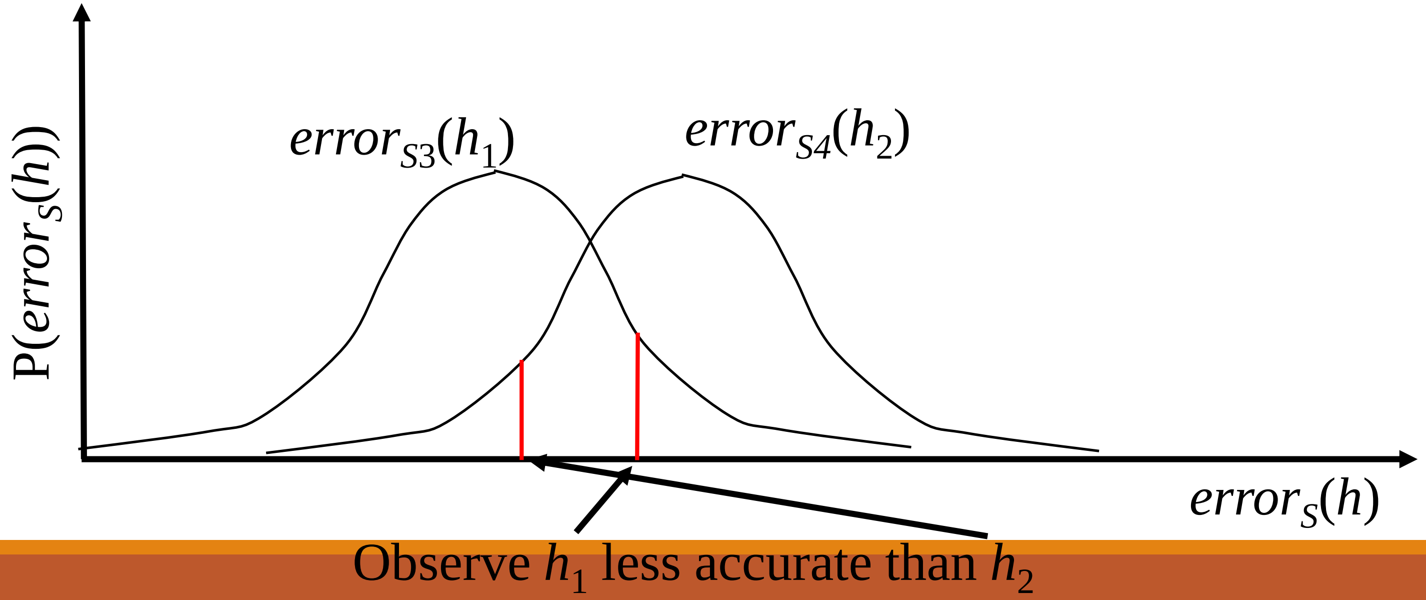
- Assume h_1 is tested on the test-set S_1 of size n_1
- Assume h_2 is tested on the test-set S_2 of size n_2



Comparing Two Learned Hypotheses

When evaluating two hypotheses (e.g. using different hyper-parameters on the same ML algorithm), their observed ordering concerning accuracy **may or may not** reflect the ordering of their **true** accuracies.

- Assume h_1 is tested on the test-set S_3 of size n_1
- Assume h_2 is tested on the test-set S_4 of size n_2



Testing alternative hypotheses

When we wish to understand **how much we can rely on a statistical finding** (e.g., that a model *h2* **is more precise than** *h1* on a **sample** dataset), we need **to list the alternatives** (e.g. **h2 in not more precise than h1** on the **entire** population).

- One of these alternatives is called the **Null Hypothesis H0**
- Usually, the **null hypothesis disconfirms** our findings

Alternative Hypothesis Tests

Suppose we measured the error rate of h1 and h2 finding that
$d = error_{S1}(h1) - error_{S2}(h2) \neq 0$**; we can perform 3 different tests:**

1. **Two-Tailed Test:** We formulate and test two alternatives:
   - ➢ **H0 (null hypothesis):** data **do not support** that **h1≠h2**
     (hence $error_{\mathcal{D}}(h1) - error_{\mathcal{D}}(h2)$ could actually be 0)
   - ➢ **H1:** data **support** that **h1≠h2**
     (d is either positive or negative; **with high confidence our finding is true; like N=95%)**

Alternative Hypothesis Tests

2. One-tailed right-test (d>0)
   - ➢ H0 (null hypothesis): data do **not support** that **h2>h1**
   - ➢ H1: data support h2>h1 (error of h1 is significantly lower)

3. One-tailed left-test (d<0)
   - ➢ H0 (null hypothesis): data do **not support** that **h2<h1**
   - ➢ H1: data support h1>h2 (error of h1 is significantly higher)

How to do: Two-Tailed Test

1. Assumes:
   - ➢ h1 is tested on the test-set S_1 of size n_1
   - ➢ h2 is tested on the test-set S_2 of size n_2
   - ➢ n_1>30, n_2 >30 to hold the Central Limit Theorem
   - ➢ **Note**: for CLT, binomial is approximated by Gaussian, so both $error_{s1}(h1)$ and $error_{s2}(h2)$ approximately follow a Gaussian distribution.
2. Suppose we wish to **estimate** the difference $d_{\mathcal{D}}$ (the «true» difference) between the (unknown) true errors of these two hypotheses:

$$d_D = error_D(h1) - error_D(h2)$$

3. As usual, define d_S the **estimator** of $d_{\mathcal{D}}$ with the known sample errors:

$$d_S = error_{s1}(h1) - error_{s2}(h2)$$

   - ➢ d_S is an **unbiased estimator** of $d_{\mathcal{D}}$. We will not give the proof.

How to do: Two-Tailed Test

4. To obtain the confidence interval we need to compute the $\boldsymbol{\sigma_{ds}}$:

Note: What is the probability distribution governing the random variable? For CLT, we know that both $error_{s1}(h1)$ and $error_{s2}(h2)$ follow distributions that are approximately Gaussian. Because the difference of two Normal distributions (Gaussian) **is also a Normal distribution**, d_S will also follow an approximately Normal distribution, with mean $d_{\mathcal{D}}$ and variance:

$$VAR(d_S) \approx \frac{error_{s1}(h1) \cdot (1 - error_{s1}(h1))}{n_1} + \frac{error_{s2}(h2) \cdot (1 - error_{s2}(h2))}{n_2}$$

Note: It can also be shown that the **variance of this distribution is the sum of the variances of $error_{s1}(h1)$ and $error_{s2}(h2)$**

$$\sigma_{d_S} = \sqrt{VAR(d_S)} \approx \sqrt{\frac{error_{s1}(h1) \cdot (1 - error_{s1}(h1))}{n_1} + \frac{error_{s2}(h2) \cdot (1 - error_{s2}(h2))}{n_2}}$$

How to do: Two-Tailed Test

If H0 (null hypothesis) holds true, then we must have:

$error_D(h1) = error_D(h2) \Rightarrow d_D = 0$

Which means: although in our experiments we observe that $error_S(h1) \neq error_S(h2)$ (e.g., $error_S(h1) < error_S(h2)$), **the "true" expected value of error differences of h1 and h2 on $\mathcal{D}$ is zero**.

- To test the likelihood of **H0**, we have to consider:

$$d_S = error_{s1}(h1) - error_{s2}(h2)$$

$$d_D = error_D(h1) - error_D(h2) = 0$$

$$|d_S - d_D| = |d_S - 0| = |d_S| \leq z\sigma_S \Rightarrow -z\sigma_S \leq d_S \leq z\sigma_S$$

$$z = \frac{d_S}{\sigma_S}$$

Error estimates on the samples

$d_{\mathcal{D}}$ must be zero if H0 holds true

Error bounds in estimating d_S

We know both d_S and $\boldsymbol{\sigma_{ds}}$ so **we compute z** and look on a z-table, to see "how many times" our result d_S is far from the expected mean difference (which is zero according to H0)

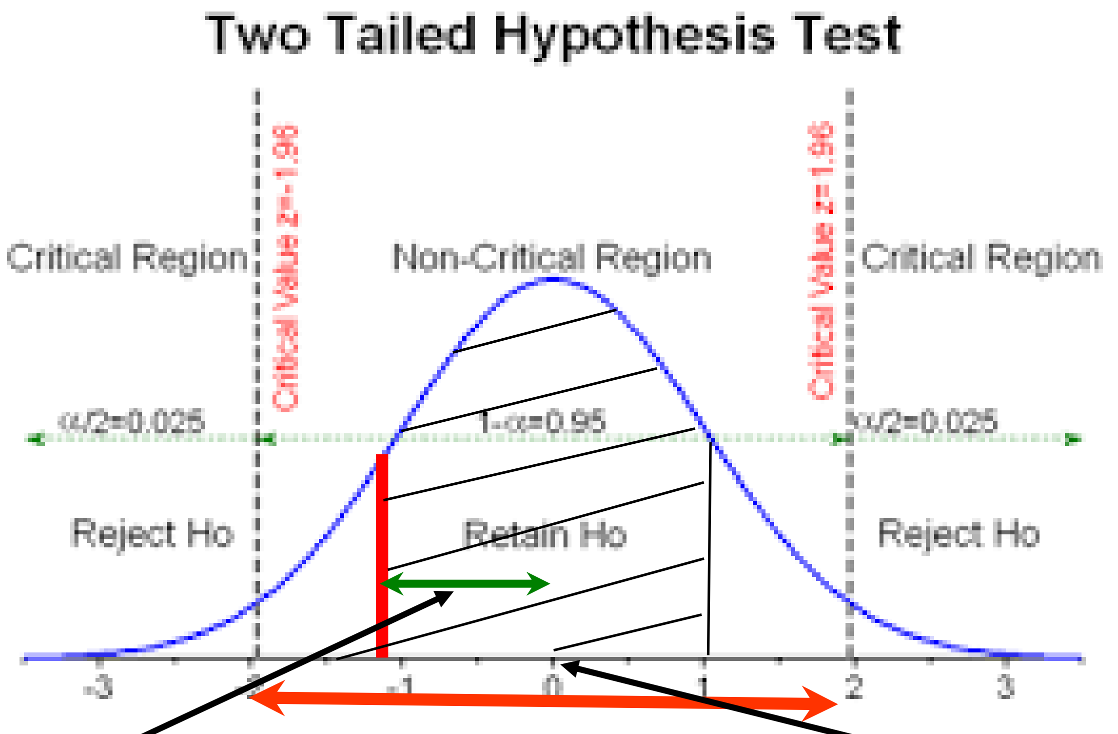
How to do: Two-Tailed Test

If the area lies **within** the non-critical region (i.e. **N≤95%**), the **Null Hypothesis H0** is **accepted** (= there is no significant difference between the two hypotheses)

$|d_S-0|=z\sigma$, $z=|d_S|/\sigma$

Given z, using the table we can compute N (the confidence area).

The "common wisdom" is that the acceptance region for H0 is within -2σ and +2σ (N≤95%)



We estimate d_S on the sample

If H0 holds, $d_D=0$

How to do: Two-Tailed Test

- In other terms: the farther our measured distance d_S is from the "expected" distance ($d_{\mathcal{D}}=0$ in case the null hypothesis H0 holds), **the less confident we should be in H0.**
- For any measured value of d_S, the y-axis gives us **the probability of observing that value**
- **If d_S is farther than ±2σ from $d_{\mathcal{D}}$,** then we may conclude that the probability of having observed the value d_S in case $d_{\mathcal{D}}=0$ is too small. And hence **we reject H0** as being very unlikely.

How to do:

Example of Two-Tailed Test

Assume:

- $d_S=0.15$
- $\sigma_S=0.05$
- $z=d_S/\sigma=3$

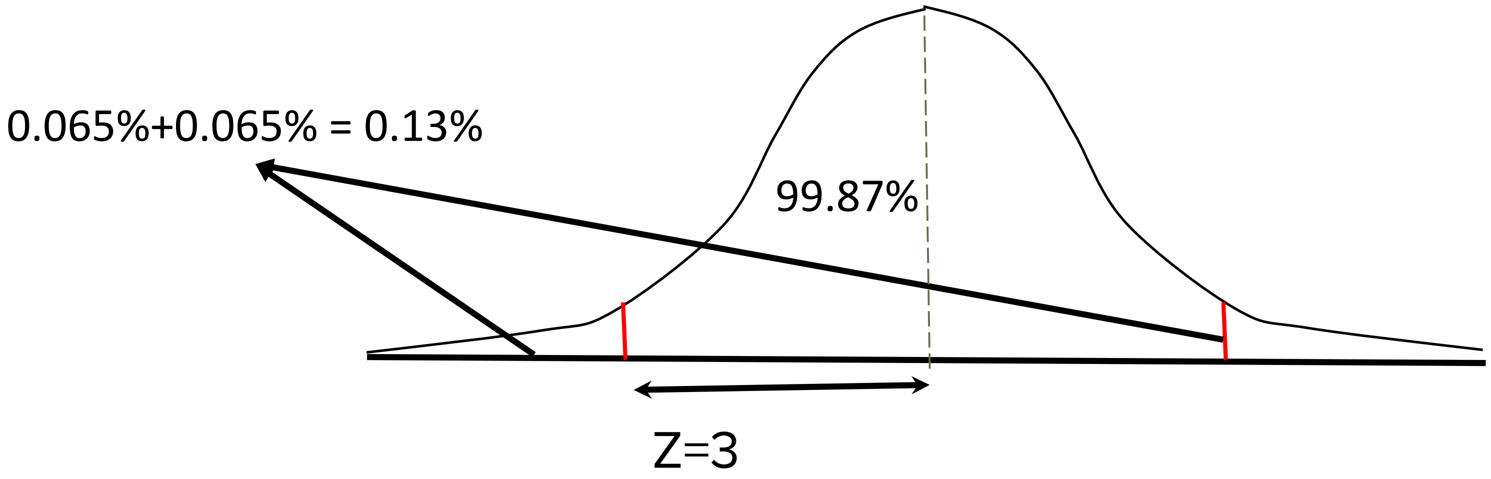
Then ⇒ N=99,87%

z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	[illegible]	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
[illegible]	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990
3.1	0.9990	0.9991	0.9991	0.9991	0.9992	0.9992	0.9992	0.9992	0.9993	0.9993
3.2	0.9993	0.9993	0.9994	0.9994	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
3.3	0.9995	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998
3.5	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998

How to do:
Example of Two-Tailed Test

Our z-test says that, if $d_{\mathcal{D}}$=0, the **probability** to obtain the value d_S=0.15 or more, is **less than 0.13%** (100-99,87)!! **So H0 is very unlikely**

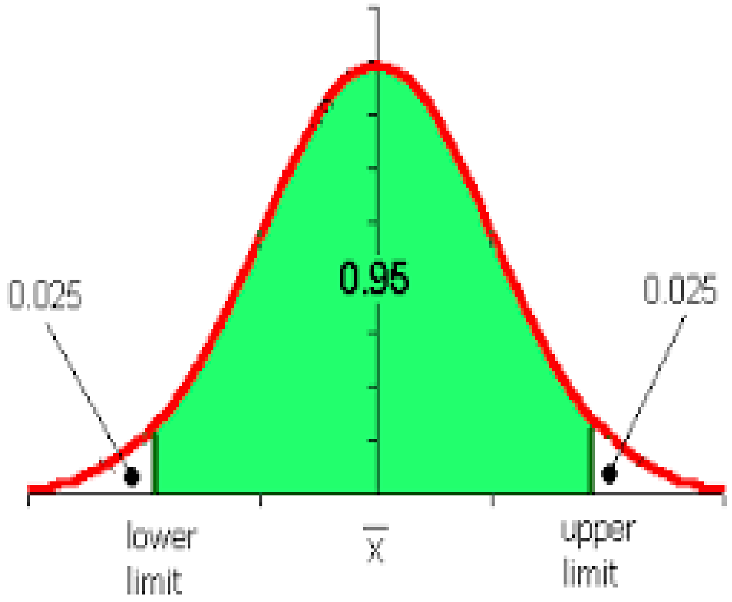
We should reject H0!!



p-value

The p-value is the “probability value” of observing our estimate, **given that H0 holds.**

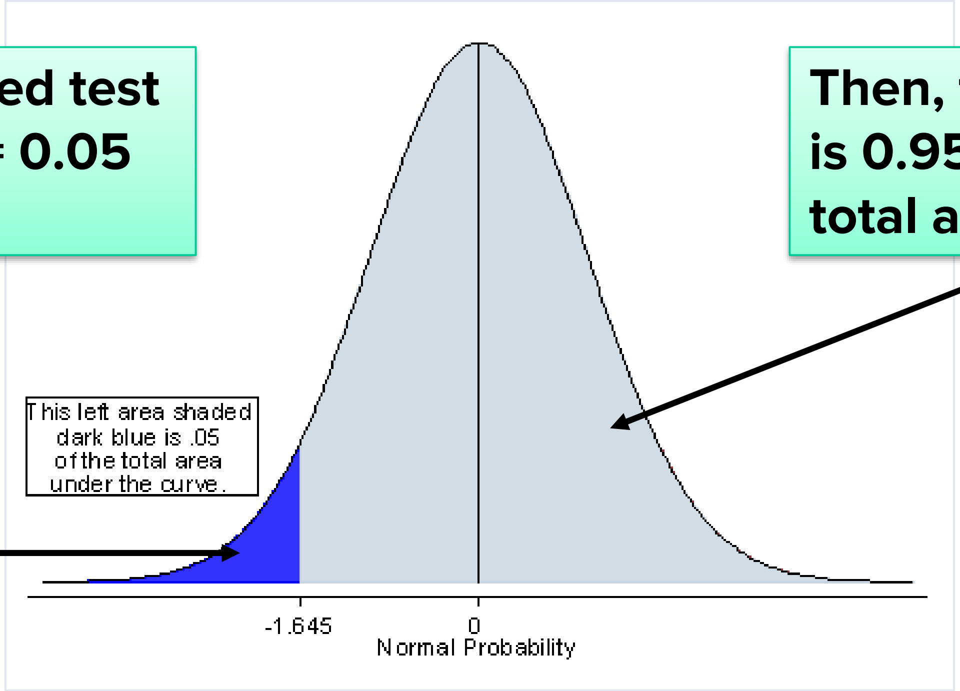
- **The common wisdom** is to **reject the null hypothesis** if **p<0.05 (5% the area under the curve of the tails)** (same as saying that the estimated value lies outside the ±2σ interval, or outside the 95% probability mass around the mean)
- In the previous example, we obtained **p** < **0.0013 (0.13%)**



How to do: One-Tailed Test

For the 1-tailed test the p-value = 0.05 (5%)

Then, the right area is 0.95 (95%) of the total area



In a **one-tailed test** we test either **h1>h2 or h1<h2**

In case **h1>h2**, we state the null hypothesis as follows:

- **H0**: **not support** that h1>h2 (hence h1≤h2)
- **H1**: **support** h1>h2 (in this case we should get an estimate of d)

How to do:

Example of **One-Right** tailed test

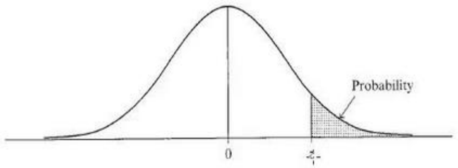
Right-tailed test (h2 > h1 (d>0)):

- $error_{s1}(h1)$= x1=17.5%, $error_{s2}(h2)$= x2=12.4%, d=5.1% (0.51)
- n_1=50, n_2=50

$$\sigma_{d_S} \approx \sqrt{\frac{error_{s1}(h1)\cdot(1-error_{s1}(h1))}{n_1} + \frac{error_{s2}(h2)\cdot(1-error_{s2}(h2))}{n_2}}$$

$$= \sqrt{\frac{0.175\cdot(1-0.175)}{50} + \frac{0.124\cdot(1-0.124)}{50}} = \sqrt{0.005}$$

$$z = \frac{(x1-x2)+(error_D(h_1)-error_D(h_2))}{0.07} = (0.051-0)/0.07 = 0.73$$



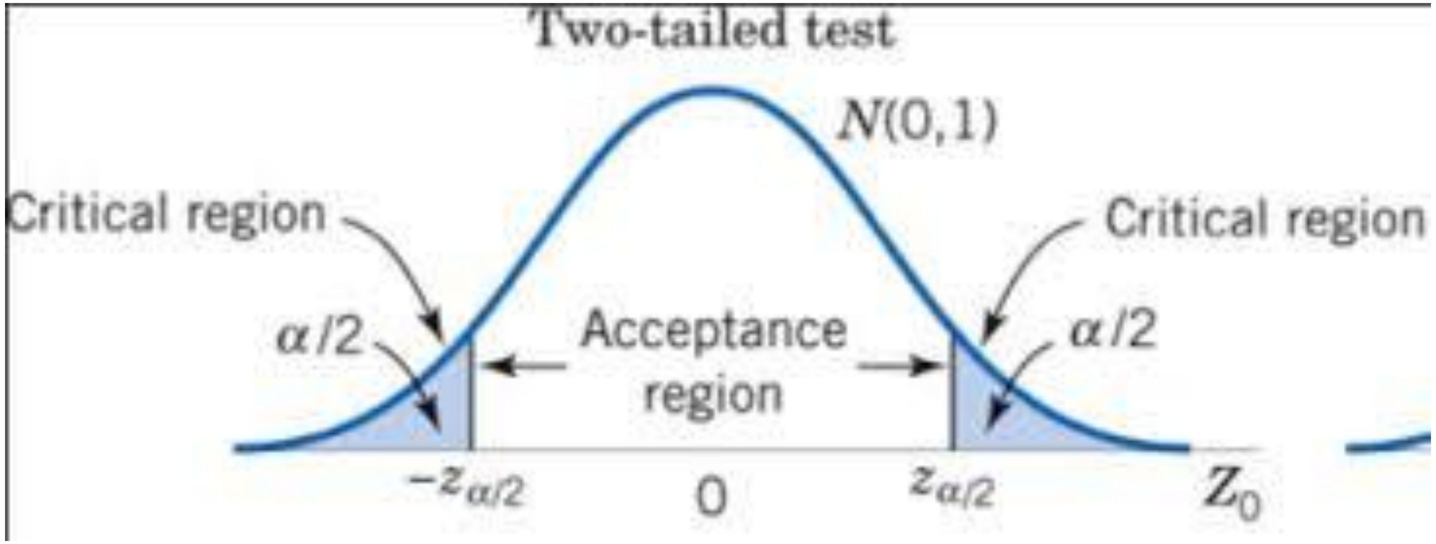
z	Second Decimal Place of z .00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	.5000	.4960	.4920	[illegible]	.4840	.4801	.4761	.4721	.4681	.4641
0.1	.4602	.4562	.4522	[illegible]	.4443	.4404	.4364	.4325	.4286	.4247
0.2	.4207	.4168	.4129	[illegible]	.4052	.4013	.3974	.3936	.3897	.3859
0.3	.3821	.3783	.3745	[illegible]	.3669	.3632	.3594	.3557	.3520	.3483
0.4	.3446	.3409	.3372	[illegible]	.3300	.3264	.3228	.3192	.3156	.3121
0.5	.3085	.3050	.3015	[illegible]	.2946	.2912	.2877	.2843	.2810	.2776
0.6	.2743	.2709	[illegible]	.2643	.2611	.2578	.2546	.2514	.2483	.2451
0.7	[illegible]	[illegible]	[illegible]	$\boxed{.2327}$	.2296	.2266	.2236	.2206	.2177	.2148
0.8	.2119	[illegible]	.2061	[illegible]	.2005	.1977	.1949	.1922	.1894	.1867
0.9	.1841	.1814	.1788	.1762	.1736	.1711	.1685	.1660	.1635	.1611
1.0	.1587	.1562	.1539	.1515	.1492	.1469	.1446	.1423	.1401	.1379
1.1	.1357	.1335	.1314	.1292	.1271	.1251	.1230	.1210	.1190	.1170

N=0.2327 = 23.27%⇒ p>0.05

The null hypothesis is **accepted**:

- The difference is not large enough to support h1<h2 (p **is not lower** than 0.05)

Summary:
Two-Tailed Test



Comparing Two Learning Algorithms

- Comparing the average accuracy of hypotheses produced **by two different ML algorithms** is more difficult. Ideally, we want to measure:

$$E_{S \subset D}(error_D(L_A(S)) - error_D(L_B(S)))$$

  - where $L_X(S)$ represents the hypothesis learned by learning algorithm L_X from training data S.
- To accurately estimate this, we need to average over **multiple, independent training and test sets**.
- However, since labeled data is limited, generally must average over multiple splits of the overall data set into training and test sets (**K-Fold Cross Validation,** see the beginning of this lesson).

How to use:

K-Fold Cross Validation to evaluate different learning algorithms

Randomly partition dataset D into k disjoint equal-sized (N) subsets $P_1 \ldots P_k$

For i from 1 to k do:

Use P_i for the test set and remaining data for training

$D_i = (D - P_i)$

$h_A = L_A(D_i)$

$h_B = L_B(D_i)$ (learn models on D_i)

$\delta_i = \text{error}_{P_i}(h_A) - \text{error}_{P_i}(h_B)$ (test models on P_i and compute difference)

Return the average difference in error:

$$\bar{\delta} = \frac{1}{k}\sum_{i=1}^{k}\delta_i$$

Error bound is computed as:

$$\begin{cases} \bar{\delta} \pm Z \cdot \sigma_{\bar{\delta}} \\ \sigma_{\bar{\delta}} = \sqrt{\dfrac{1}{k(k-1)}\sum_{i=1}^{k}\left(\delta_i - \bar{\delta}\right)^2} \end{cases}$$

Is L_A better than L_B?

- K-fold cross-validation improves confidence in our estimate of δ since we are performing many experiments and computing δ as the **average** of δ_i.
- As K grows this average tends to the true mean difference (however we cannot make K too big since individual samples should be large enough for the CLT to apply)
- We can, in any case, apply hypothesis testing as before

Example:
Sample Experimental Results

Which experiment provides better evidence that SystemA is better than SystemB?

Experiment 1

	SystemA	SystemB	δ
Trial 1	87%	82%	+5%
Trail 2	83%	78%	+5%
Trial 3	88%	83%	+5%
Trial 4	82%	77%	+5%
Trial 5	85%	80%	+5%
Average	85%	80%	**+5%**

Experiment 2

	SystemA	SystemB	δ
Trial 1	90%	82%	+8%
Trail 2	93%	76%	+17%
Trial 3	80%	85%	–5%
Trial 4	85%	75%	+10%
Trial 5	77%	82%	– 5%
Average	85%	80%	**+5%**

Experiment 1 mean δ has σ=0, therefore we have perfect confidence in the estimate of δ

Experimental Evaluation: Conclusions

- Good experimental methodology is important for evaluating learning methods.
- Important to test on a variety of domains to demonstrate generality for a variety of problems. Testing on 10+ data sets is common.
- Variety of freely available data sources
  - ➢ UCI Machine Learning Repository (link)
  - ➢ KDD Cup (large data sets for data mining) (link)
  - ➢ CoNLL Shared Task (natural language problems)(link)
- Data for real problems is preferable to artificial problems to demonstrate usefulness in real contexts.
- Many available datasets have been subjected to significant feature engineering to make them learnable.

Related links

- Metrics: link, link, link, link, link
- Basic statistics: link
- Bias, Variance, and Error: link, link, link
- Estimator: link, link
- Estimating the accuracy of a hypothesis/Confidence Interval:
  - Text: link, link, link, link, link, link
  - Video: link
  - Hypothesis testing: link, link, link
  - Z-table: link