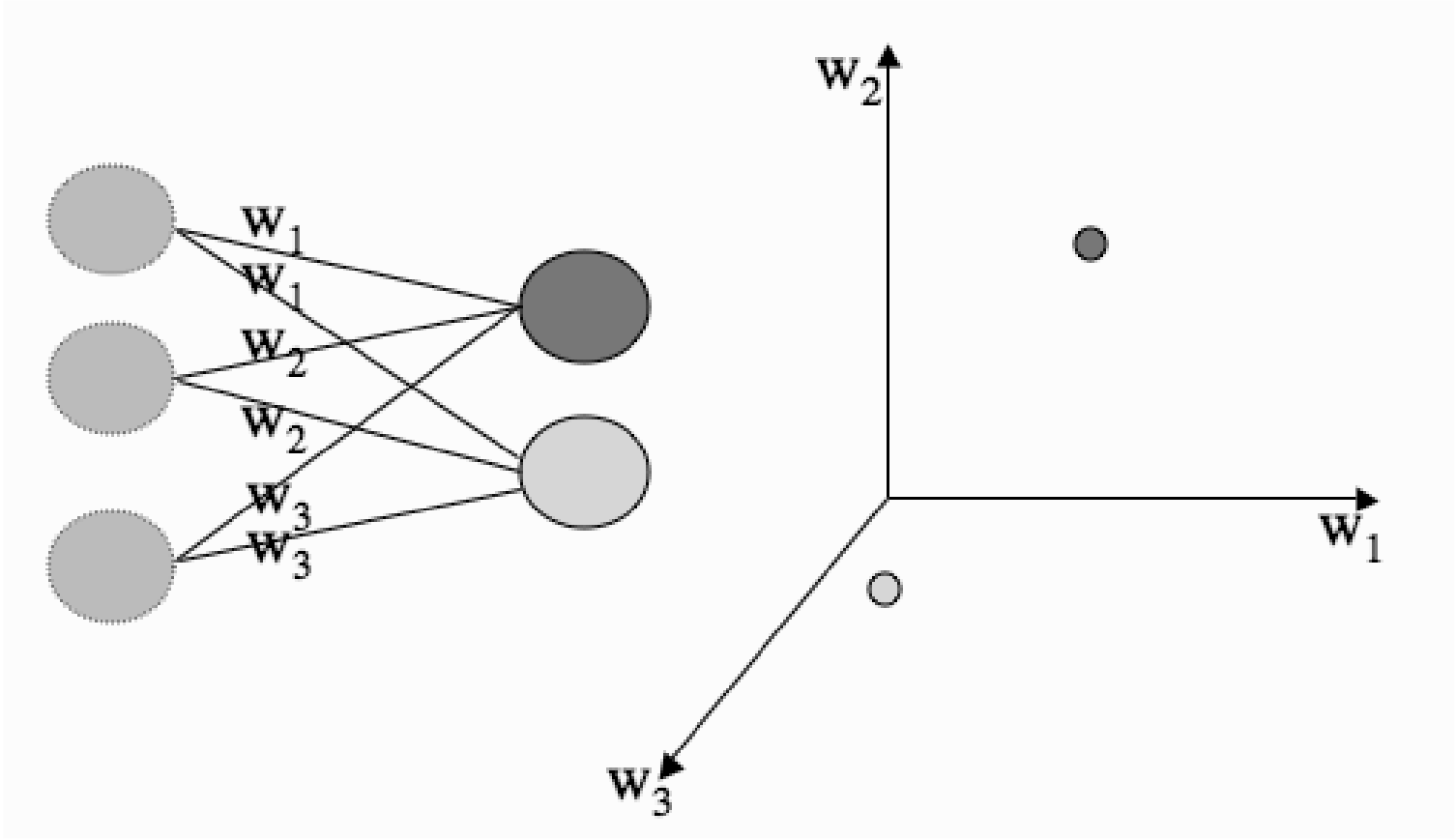
ML Prelimenaries

SOME TERMINOLOGY

We will often write equations in vector and matrix notation, with lowercase boldface letters being used for vectors and uppercase boldface letters for matrices. A vector **x** has elements (*x*1*, x*2*, . . . , xm*). We will use the following notation in the book:

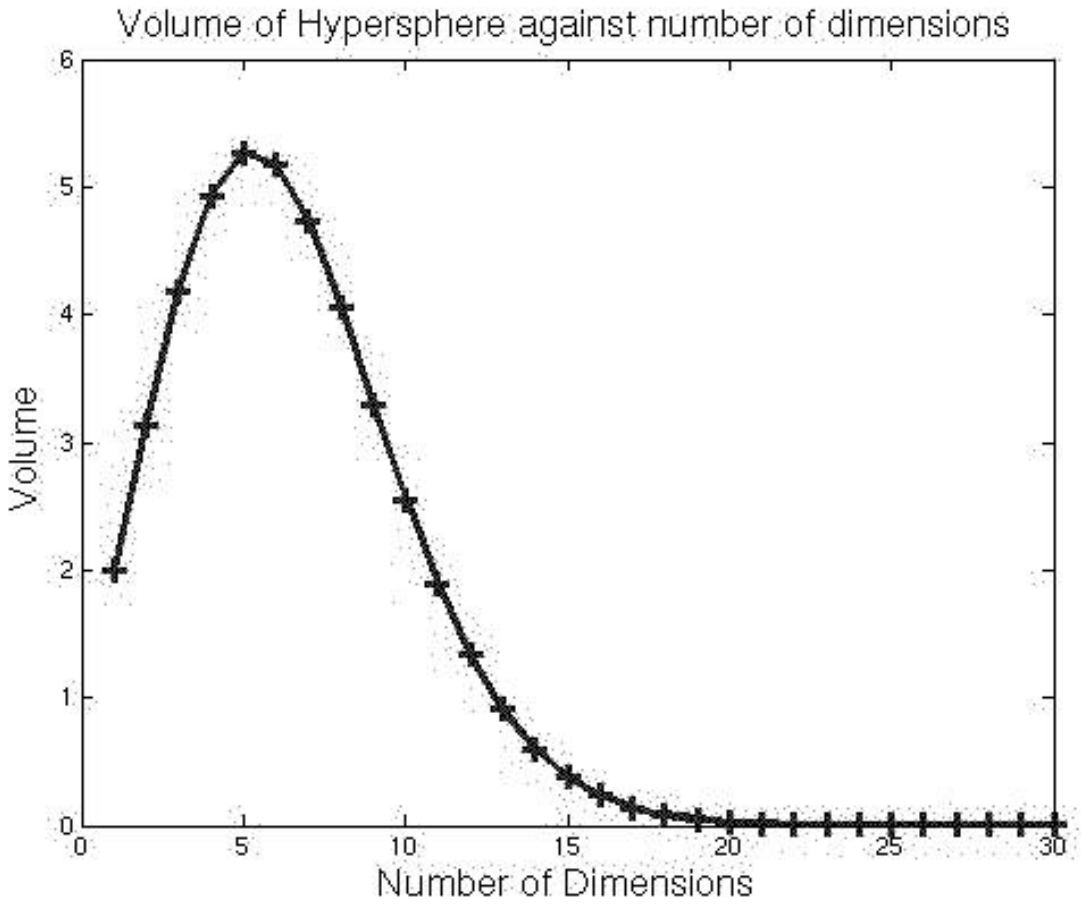
- **Inputs** An input vector is the data given as one input to the algorithm. Written as **x**, with elements *xi*, where *i* runs from 1 to the number of input dimensions, *m*.
- **Weights** *wij* , are the weighted connections between nodes *i* and *j*. For neural networks these weights are analogous to the synapses in the brain. They are arranged into a matrix **W**.
- **Outputs** The output vector is **y**, with elements *yj* , where *j* runs from 1 to the number of output dimensions, *n*. We can write **y**(**x***,***W**) to remind ourselves that the output depends on the inputs to the algorithm and the current set of weights of the network.
- **Targets** The target vector **t**, with elements *tj* , where *j* runs from 1 to the number of output dimensions, *n*, are the extra data that we need for supervised learning, since they provide the 'correct' answers that the algorithm is learning about.
- **Activation Function** For neural networks, $g(\cdot)$ is a mathematical function that describes the firing of the neuron as a response to the weighted inputs.
- **Error** *E*, a function that computes the inaccuracies of the network as a function of the outputs **y** and targets **t**.

Weight space - Neuron position in weight space



The Curse of Dimensionality - Dimensions vs volume

Dimension	Volume
1	2.0000
2	3.1416
3	4.1888
4	4.9348
5	5.2636
6	5.1677
7	4.7248
8	4.0587
9	3.2985
10	2.5502



- It was computed using the formula for the volume of the hypersphere of dimension n as $v_n = (2\pi/n)v_{n-2}$.

So as soon as $n > 2\pi$, the volume starts to shrink.

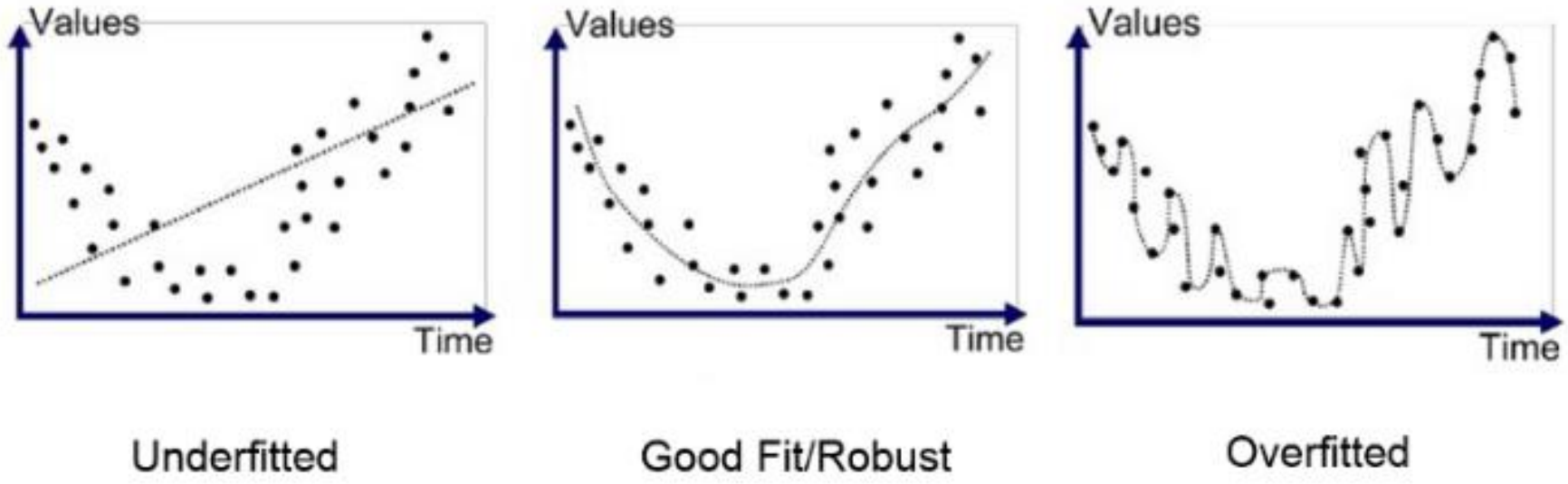
Note: As the number of dimensions increases, the volume of the unit hypersphere does not increase with it

- The curse of dimensionality applies to machine learning algorithms because as the number of input dimensions gets larger, the algorithm needs more data to enable the algorithm to generalise sufficiently well

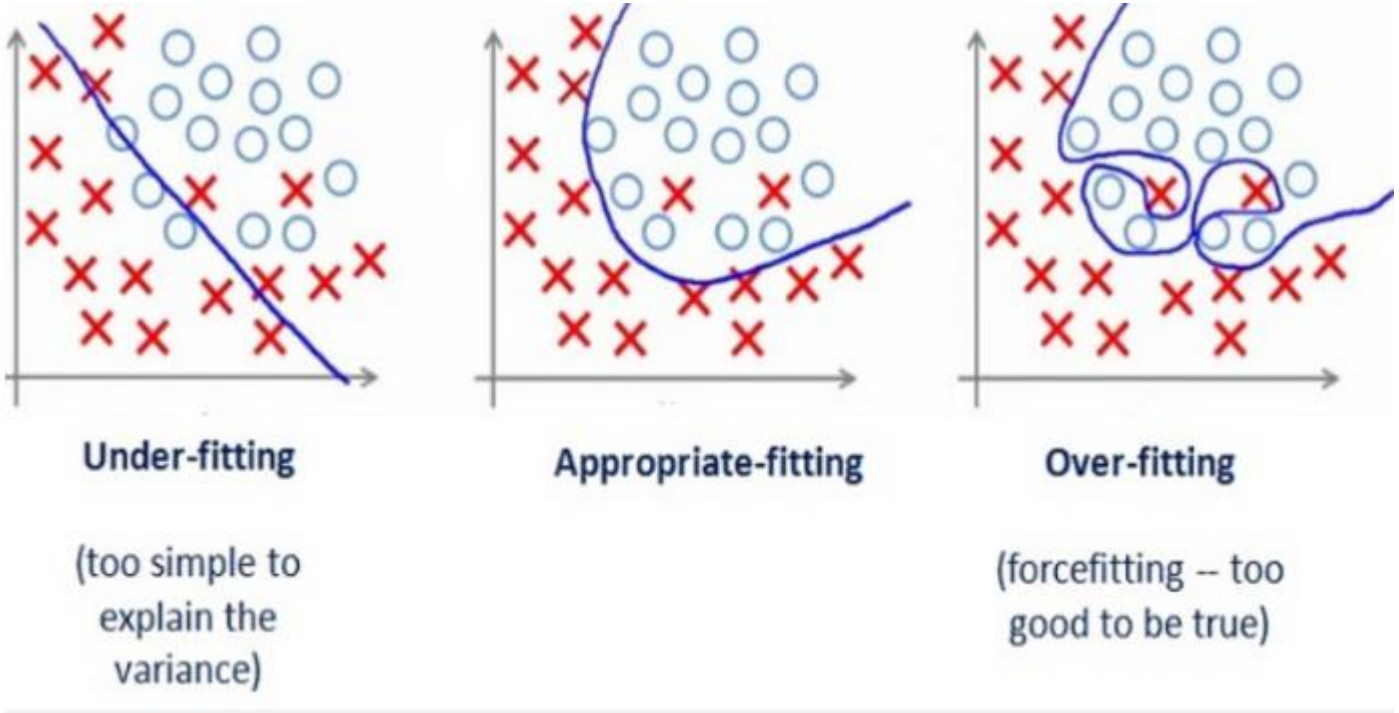
TESTING MACHINE LEARNING ALGORITHMS – overfitting and under fitting

Overfitting refers to a model that models the training data too well.
Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.
Overfitting is more likely with nonparametric and nonlinear models that have more flexibility when learning a target function.

Underfitting refers to a model that can neither model the training data nor generalize to new data.
An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data.



overfitting and under fitting contd..



Training, Testing, and Validation Sets

- ***Training set**: A set of examples used for learning, that is to fit the parameters of the classifier.*
- ***Validation set**: A set of examples used to tune the parameters of a classifier, for example to choose the number of hidden units in a neural network.*
- ***Test set:** A set of examples used only to assess the performance of a fully-specified classifier.*

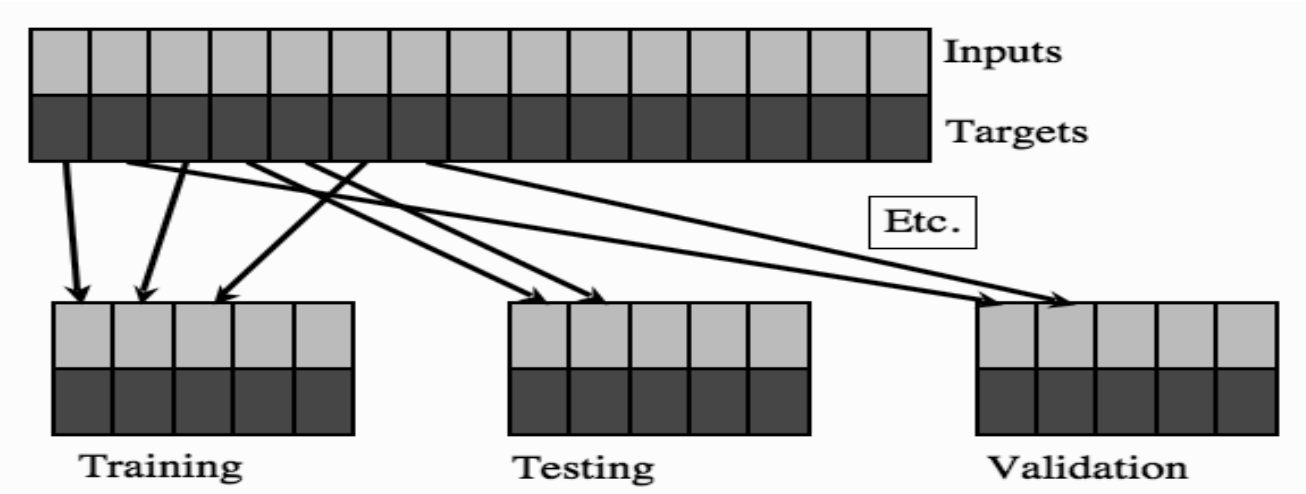
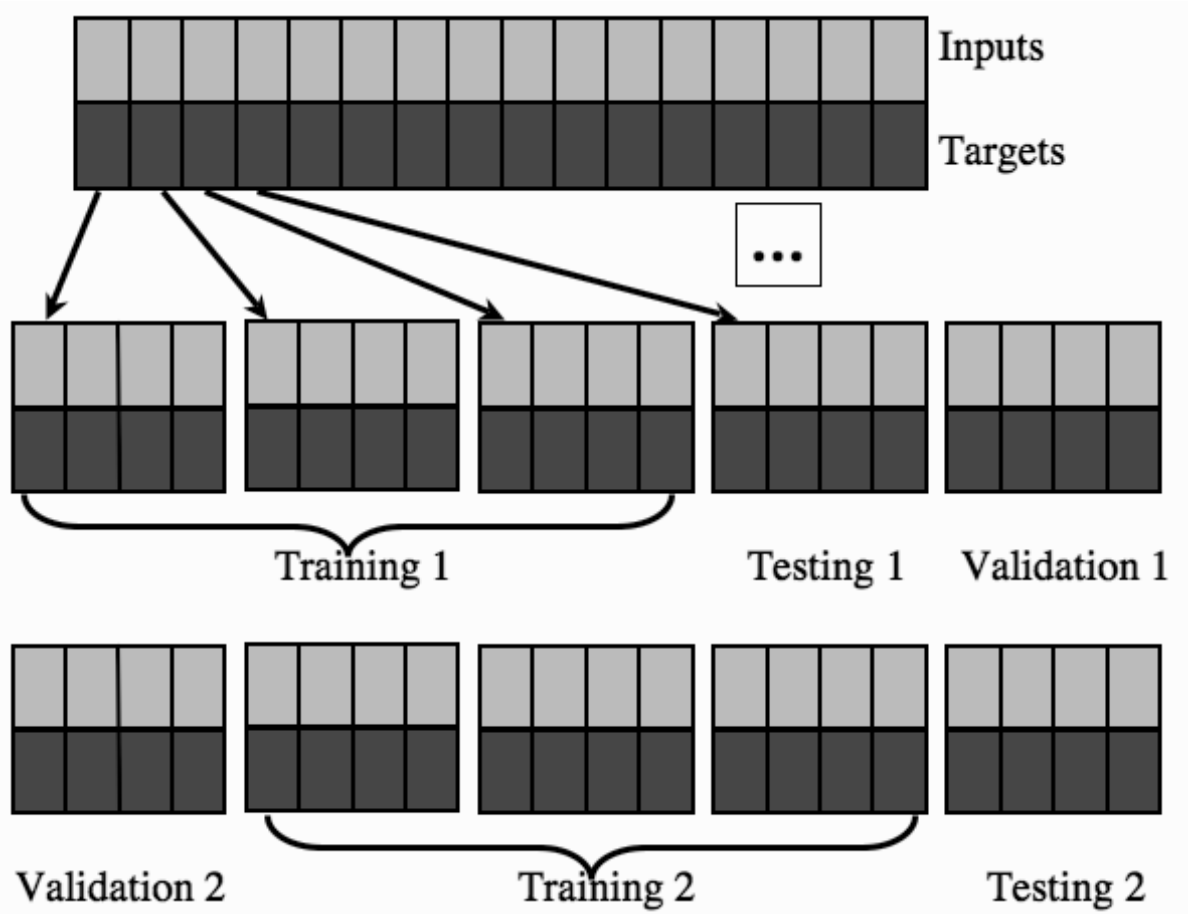


Figure: The dataset is split into different sets, some for training, some for validation, and some for testing

Training, Testing, and Validation Sets contd..



- Generally, the exact proportion of training to testing to validation data is up to you, but it is typical to do something like **50:25:25** if you have plenty of data, and **60:20:20** if you don't
- If you are really short of training data, so that if you have a separate validation set there is a worry that the algorithm won't be sufficiently trained; then it is possible to cross-validation perform **leave-some-out, multi-fold cross-validation**

Figure: Leave-some-out, multi-fold cross-validation gets around the problem of data shortage by training many models. It works by splitting the data into sets, training a model on most sets and holding one out for validation (and another for testing). Different models are trained with different sets being held out.

Confusion matrix

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

What can we learn from this matrix?

- There are two possible predicted classes: "yes" and "no". If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease.
- The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).
- Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.
- In reality, 105 patients in the sample have the disease, and 60 patients do not.

Confusion matrix

- **true positives (TP):** These are cases in which we predicted yes (they have the disease), and they do have the disease.
- **true negatives (TN):** We predicted no, and they don't have the disease.
- **false positives (FP):** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- **false negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

Accuracy Metrics

- **Accuracy:** Overall, how often is the classifier correct?
  - (TP+TN)/total = (100+50)/165 = 0.91
- **Misclassification Rate:** Overall, how often is it wrong?
  - (FP+FN)/total = (10+5)/165 = 0.09
  - equivalent to 1 minus Accuracy
  - also known as "**Error Rate**"
- **True Positive Rate:** When it's actually yes, how often does it predict yes?
  - TP/actual yes = 100/105 = 0.95
  - also known as "**Sensitivity**" or "Recall"
- **False Positive Rate:** When it's actually no, how often does it predict yes?
  - FP/actual no = 10/60 = 0.17
- **True Negative Rate:** When it's actually no, how often does it predict no?
  - TN/actual no = 50/60 = 0.83
  - equivalent to 1 minus False Positive Rate
  - also known as "**Specificity**“

Accuracy Metrics contd..

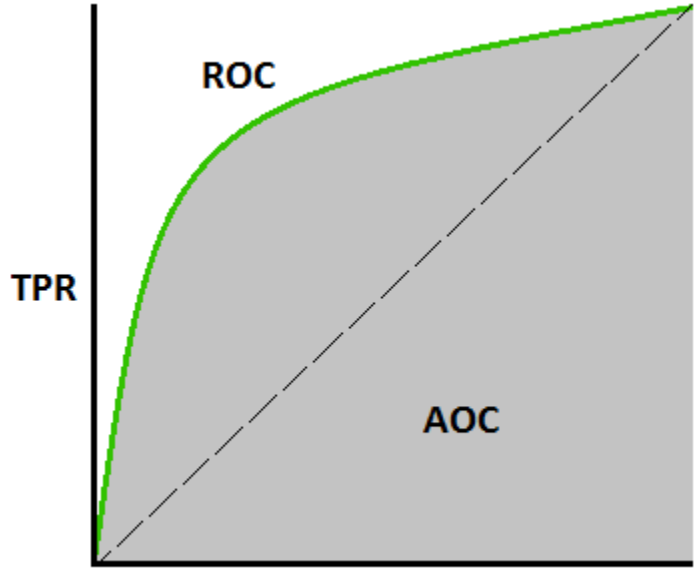
- **Precision:** When it predicts yes, how often is it correct?
  - TP/predicted yes = 100/110 = 0.91
- **Prevalence:** How often does the yes condition actually occur in our sample?
  - actual yes/total = 105/165 = 0.64
- **F Score:** This is a weighted average of the true positive rate (recall) and precision.

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

$$F_1 = \frac{\#TP}{\#TP + (\#FN + \#FP)/2}.$$

ROC (Receiver Operator Characteristic) Curve

- It is a commonly used graph that summarizes the performance of a classifier over all possible thresholds.
- It is generated by plotting the True Positive Rate (y-axis) against the False Positive Rate (x-axis) as you vary the threshold for assigning observations to a given class.



Unbalanced datasets

- In the case where the dataset is not balanced, the balanced accuracy can be omputed as the sum of sensitivity and specificity divided by 2.

- But the correct measure is Matthew's Correlation Coefficient provides a balanced accuracy computation, which is computed as:

$$MCC = \frac{\#TP \times \#TN - \#FP \times \#FN}{\sqrt{(\#TP + \#FP)(\#TP + \#FN)(\#TN + \#FP)(\#TN + \#FN)}}$$

If any of the brackets in the denominator are 0, then the whole of the denominator is set to 1.

these methods of evaluation, if there are more than two classes and it is useful to distinguish the different types of error, then the calculations get a little more complicated

Measurement Precision

Precision

- The measure of the variability of the algorithm is also known as **precision**
- tells how repeatable the predictions are made by the algorithm
- similar to the variance of a probability distribution - it tells you how much spread around the mean to expect

Trueness

- the average distance between the correct output and the prediction
- Trueness doesn't usually make much sense for classification problems unless there is some concept of certain classes being similar to each other

Measurement Precision contd..

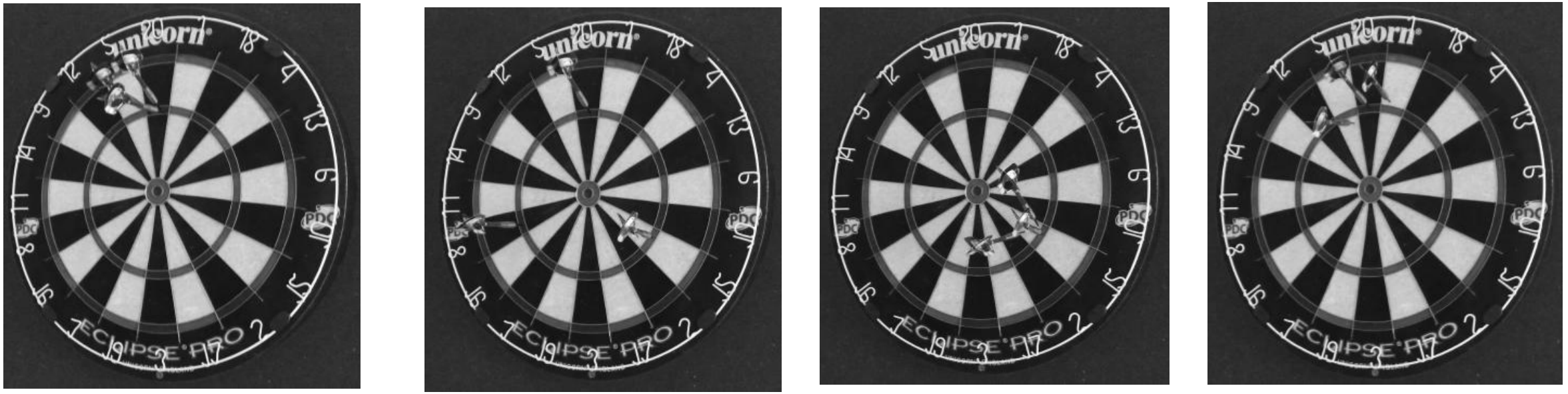


FIGURE: Assuming that the player was aiming for the highest-scoring triple 20 in darts (the segments each score the number they are labelled with, the narrow band on the outside of the circle scores double and the narrow band halfway in scores triple; the outer and inner 'bullseye' at the centre score 25 and 50, respectively), these four pictures show different outcomes. Top left: very accurate: high precision and trueness, top right: low precision, but good trueness, bottom left: high precision, but low trueness, and bottom right: reasonable trueness and precision, but the actual outputs are not very good


Thank
you