# Neural Computing

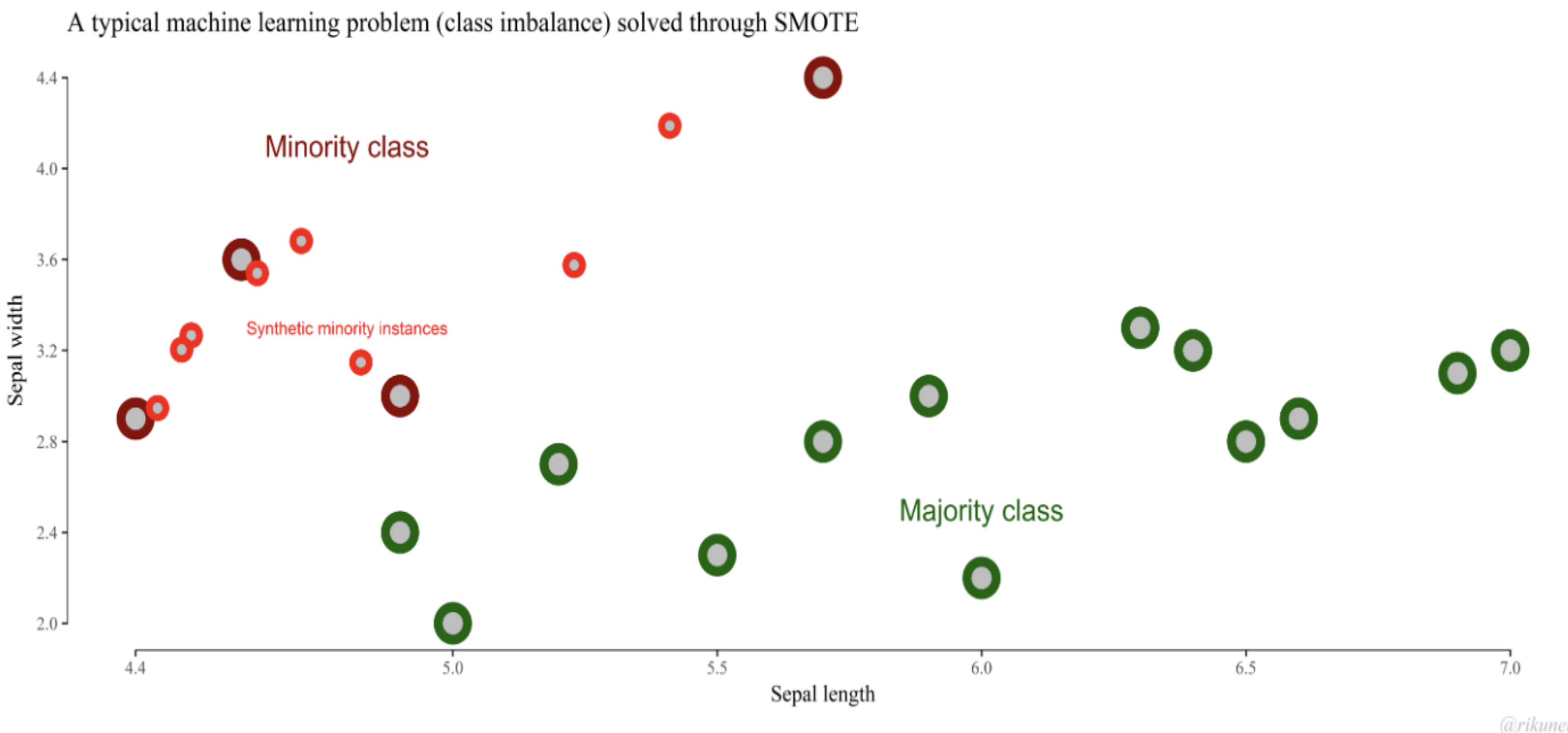
## Tutorial 3

## Supporting Notes on:

## Smote, Early Stopping and Boosting

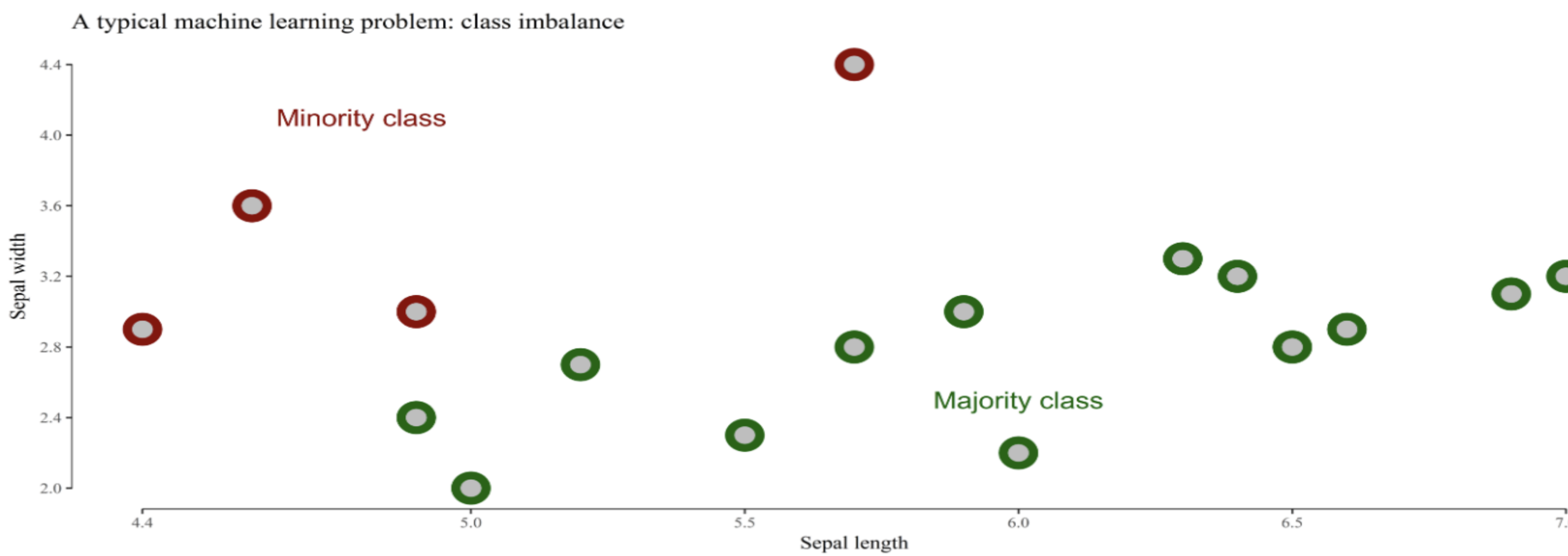
# SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) – to use when one class in the training set dominates the other by having far more examples e.g. credit card fraud prevention: 99% of transactions are genuine



# SMOTE (cont.)

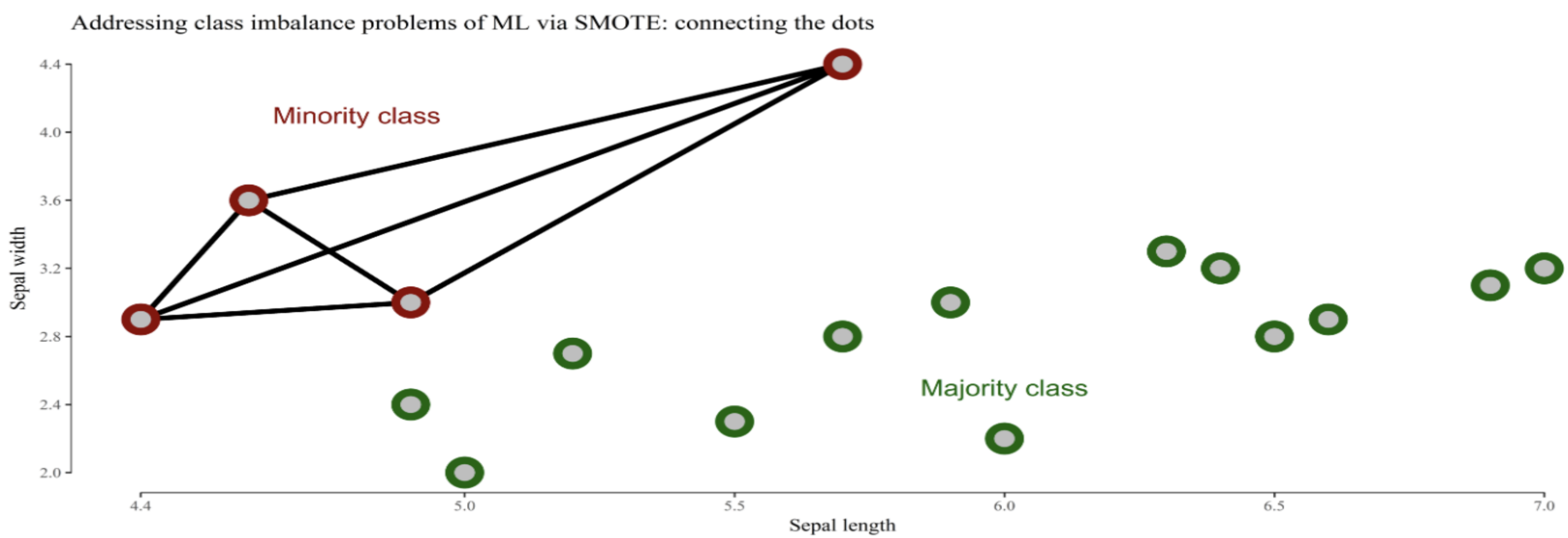
- Example: classifying iris flowers based on sepal length and width.




- You can see the setosa flowers in red (top left) and the versicolor flowers in green (bottom right). There are only 4 examples of setosa in the training sample. An easy (but undesirable) way for a learning algorithm to achieve high accuracy in this case of class imbalance is simply to classify every example as versicolor (or every card transaction as genuine for 99% accuracy!).

# SMOTE (cont.)

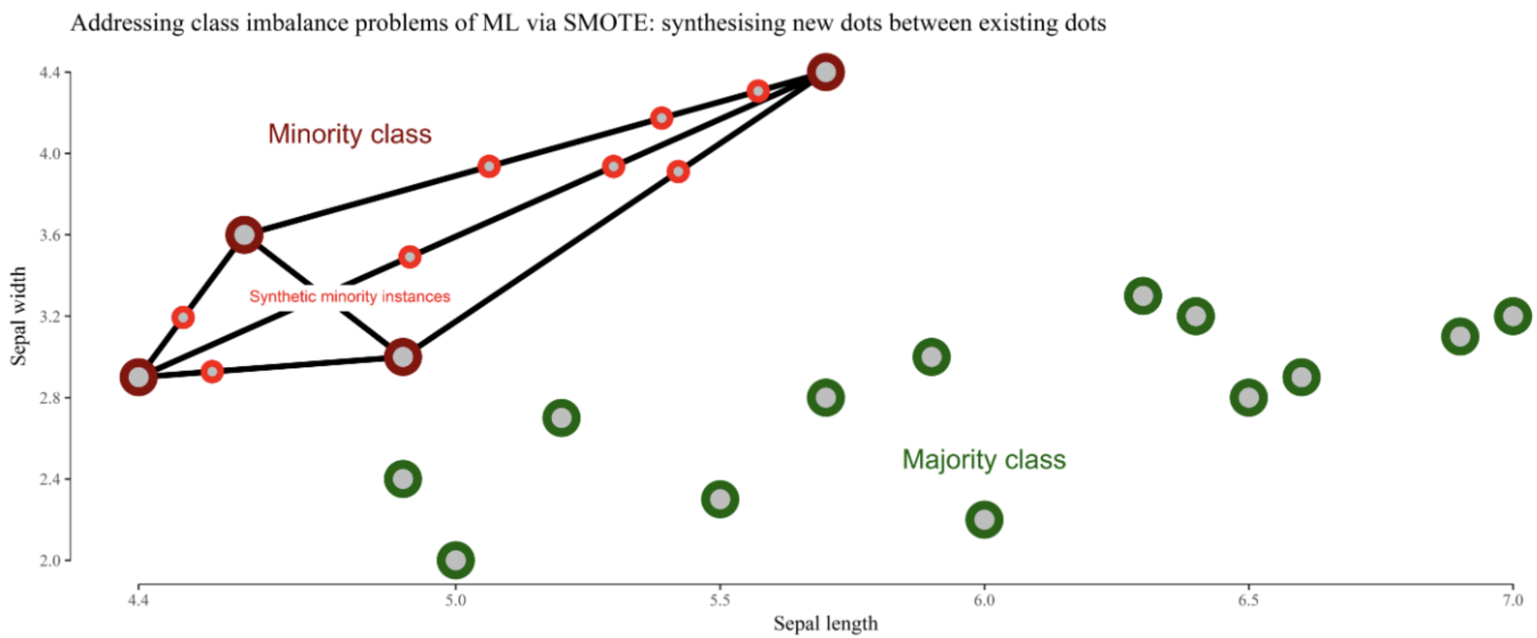
- SMOTE synthesises new minority instances between existing (real) minority instances (i.e. examples). Imagine that SMOTE draws lines between existing minority instances like this:




- SMOTE then creates new minority instances (synthetic data) somewhere on these lines.
- Think about how to make this work for more than two variables!

# SMOTE (cont.)

- After synthesising new minority instances, the imbalance shrinks from 4 red versus 13 green to 12 red versus 13 green. Red now dominates within the range of values typical for setosa on both axes (while green dominates otherwise).
- Is it a good idea to create a third “neither” class?



# Early stopping:
# Training for how many epochs?

- Too few epochs and we might underfit (i.e. not learn everything we can from the training data)

- Too many epochs and we might overfit (i.e. fit the noise in the training data rather than the signal, thus *memorizing* rather than *generalizing*).

The idea behind early stopping:

- Split data into training and test sets
- At the end of each epoch (or every *n* epochs or a *batch* of examples):

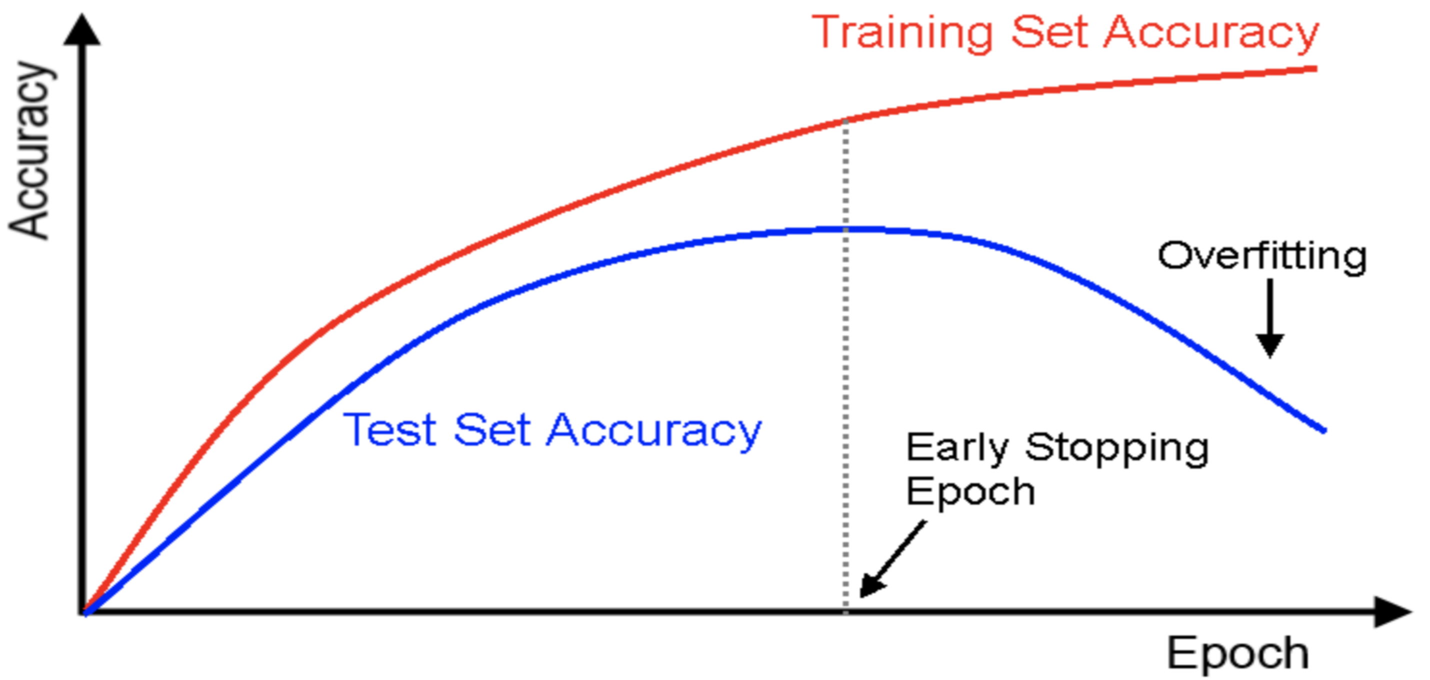
  - evaluate the network performance on the test set (i.e. without changing the weights otherwise you're data snooping!)

  - if the network outperforms the previous best model: save a copy of the network at the current epoch
- Take as your trained network the model that has the best test set performance (in other words, stop training as soon as the test set performance decreases)

# Early stopping (cont.)

The best model is the one saved at the time of the vertical dotted line (notice that the graph shows accuracy rather than error)

As soon as (or a few epochs after) the test set error increases (e.g. after 10 epochs without improvement), training stops.



# Boosting

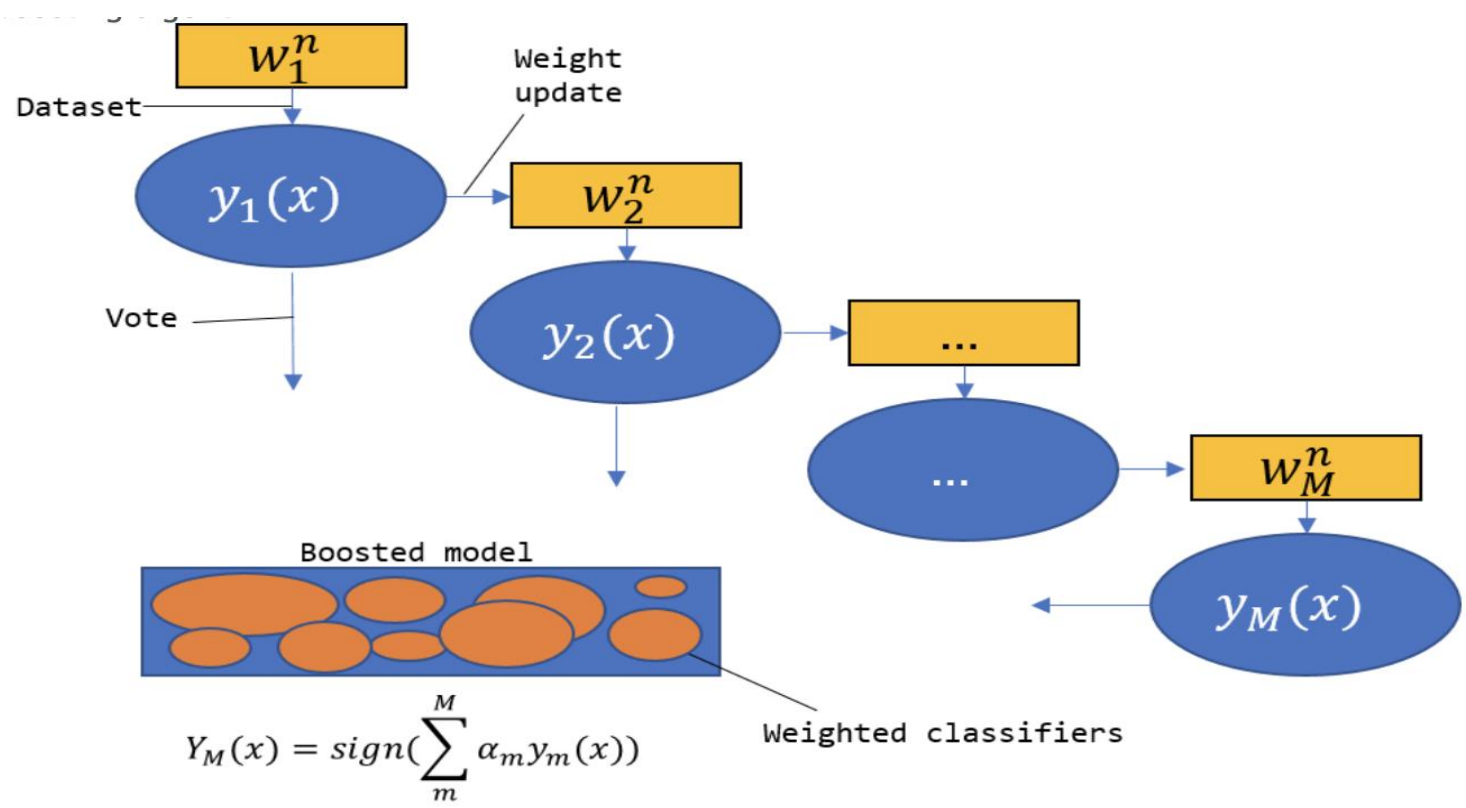
- Boosting is an ensemble technique that attempts to create a strong (i.e. good) classifier from a number of weak classifiers (i.e. not that good, but better than random).

- This is done by building a model from the training data, then creating a second model that attempts to correct the errors of the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added.

- **AdaBoost** was the first really successful boosting algorithm developed for binary classification.

# Boosting (cont.)

- **Boosting** is an **ensemble technique** that attempts to create a strong classifier from a number of weak classifiers.



# Boosting (cont.)

- The different (base) classifiers are each built on a weighted dataset where the weights of the single instances in the dataset depend on the results of the previous base classifiers for those instances.

- If they have misclassified an instance, the weight for that instance will be increased in the next model, while if the classification was correct, the weight remains the same. The *weight* here can be seen as a measure of the importance or *attention* to be paid to that example.

- The final decision making is achieved by a weighted vote from the base classifiers with those weights determined by the misclassification rates of the models.

- If a model had a high classification accuracy, it will get a high weight while it gets a low weight if it has had a poor classification accuracy.