# Section C Plan

1.

The statistical technique that I will be investigating is a machine learning algorithm called stochastic gradient descent (SGD). To explain SGD, it is easiest to first introduce this basic gradient descent (GD), its predecessor.

Gradient descent is an algorithm for learning the parameters of a machine learning model. It achieves this by finding a local minima of a cost function by iteratively adjusting each parameter so that it moves in the direction of negative gradient. This cost (or loss) function is typically an error function that measures the difference between predicted and actual y values produced by the machine learning model.

There are two main steps, an initialisation and an iteration step, involving one hyperparameter, the learning rate, normally denoted as .

Init step: Select initial parameters theta, alpha

Iter step: Update parameters using training data, gradient and alpha by following equation:

Theta i+1 = theta i -alpha x grad(theta) J(theta)

Repeat the iter step until a minima is reached.

Under basic GD, gradJ(theta) is calculated at every step against the full data set, which has a very large memory requirement. It is also susceptible to mistakes – it is very possible that it ends up in a local minima, rather than the global minima. This is dependent on our choice of starting parameters, and partially the learning rate.

A small learning rate will cause very slow convergence to minima, and will unlikely get us out of local minima, but will guarantee to find a minima where possible. A larger learning rate can converge very quickly – but can also miss minima entirely. This can be useful for escaping local minima, but has the concerning possibility of missing all minima and diverging.

SGD counters many of GDs shortcomings. It modifies GD by calculating the gradient for just one example at each iteration, massively lowering memory requirements. To remove any ordering bias, it also begins by randomly shuffling the dataset. This is the stochastic part of the algorithm – by randomly sampling data the path taken to the minima jumps over more contrasting parameter combinations, making it less likely to get stuck in a local minima.

The equation is modified as follows:

EQUATION

2.

Either

Detecting malware from file information

[https://archive.ics.uci.edu/ml/datasets/Malware+static+and+dynamic+features+VxHeaven+and+Virus+Total#](https://archive.ics.uci.edu/ml/datasets/Malware+static+and+dynamic+features+VxHeaven+and+Virus+Total)

Problem: >1000 features, data over multiple csv, sparse matrices

Predicting wine quality from chemical makeup

<https://archive.ics.uci.edu/ml/datasets/Wine+Quality>

Problem: 9 classes

Predicting hepatitis deaths from patient info

<https://archive.ics.uci.edu/ml/datasets/Hepatitis>

Problem: data not in csv

https://www.kaggle.com/fedesoriano/heart-failure-prediction

3.

Appropriate metrics:

Error insufficient?

Look at precision/recall/f1

4.

Vary alpha (learning rate), checking metrics above

5.

Cross validation to eliminate sample bias.

Aim to investigate the effects of varying the learning rate on the success of the algorithm. Since this is a healthcare dataset, the number of false positives to false negatives is important.

SVMs

Hinge loss not log loss