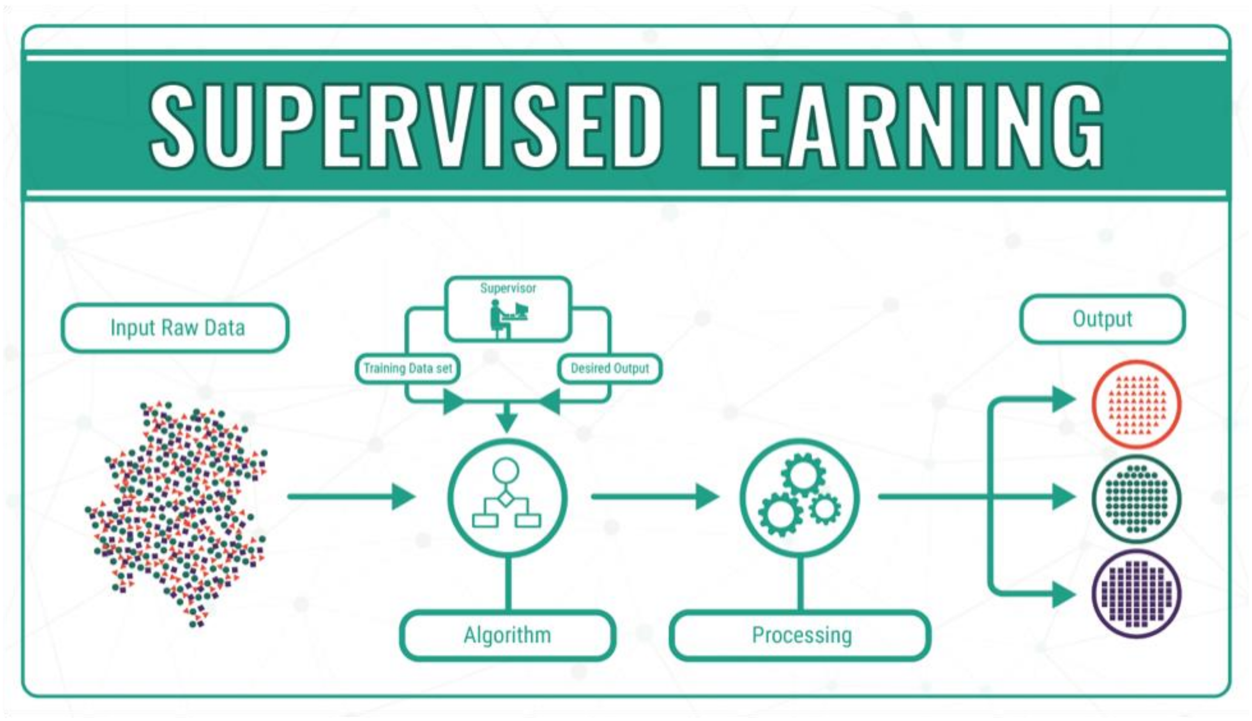
Logistic Regression Classification via the VIX

By Philippe Rivet, Electrical Engineering '19, Western University

The Chicago Board of Options Exchange's measure of implied volatility the VIX is widely perceived as an indicator of investors' estimation of market instability, and is especially relevant in times of seeming crisis. Of particular interest is whether it can be used as a 'predictor' of recessionary events when combined wit the S&P/TSX index. Typically the method of logistic regression (LR) can be leveraged to evaluate the likelihood of a binary event, but given the large volume of data available an alternative approach using machine learning (ML) methods is potentially fruitful.

Supervised Learning

Of the 3 major approaches to ML, supervised, unsupervised, and reinforcement, the 1st is perhaps the most intuitive as it involves executing computations on pre-labelled data. In this case a drop of greater than or equal to 2.5% is labelled as 'y=1' and everything else as 'y=0'. The features (i.e. vectors used to train the algorithm, conceptually similar to explanatory variables in statistics) leveraged to do this are the daily index prices and values of the VIX.



https://bigdata-madesimple.com/machine-learning-explained-understanding-supervised-unsupervised-and-reinforcement-learning/

Classification by Logistic Regression

In a nutshell, the algorithm is given a 'hypothesis' function it seeks to optimize with the end goal of producing a parameter vector. Once this process is complete this vector (typically denoted θ) can then be used to classify new values on a given dataset different than the training dataset according to the supervisor's criterion.

For simple LR the hypothesis is:

$$h_\theta(x) = g(\theta^T x)$$

Where g is the logistic or sigmoid function

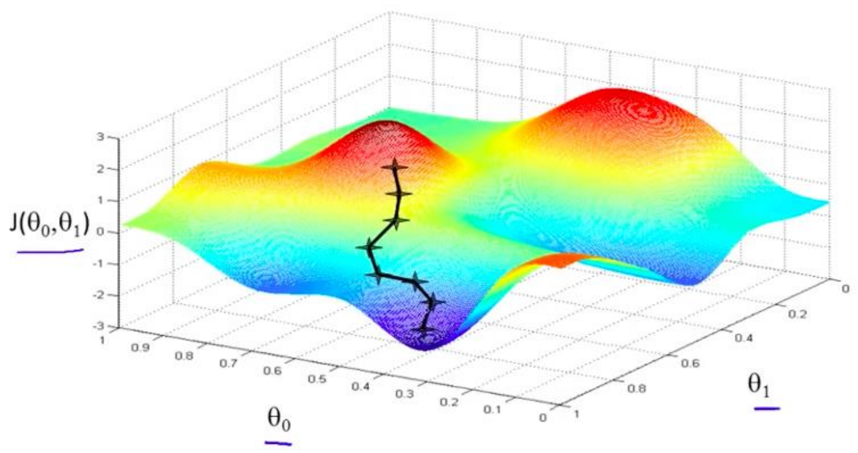
$$g(z) = \frac{1}{1 + e^{-z}}$$

In the most basic case, h_θ is a linear function so y=1 will be predicted when its value is greater than some constant, say 0.25. This will produce a straight line for a 2-D classification problem.

As more features are added, h_θ is upgraded to a polynomial form so more complicated 'curvy' decision boundaries can be produced for non-trivial classification tasks.

So how does the algorithm decide on the values of the parameters θ?

The method of gradient descent is used, as in minimize a loss function or cost function that represents desired events using real numbers. In 2 dimensions this amounts to finding the local minimum of a curve, and in 3 dimensions the objective is to find a 'valley' of a surface mapping with the feature variables:



https://stats.stackexchange.com/questions/241715/gradient-descent-and-cost-function-trouble

The intuition for the cost function is that it should satisfy two properties:

i) Be convex, loosely speaking a relatively 'smooth' curve that won't cause the optimizer to become stuck in jagged local minima

ii) Be able to predict values properly, as in

   a. Cost value is zero if y=1 and h_θ=1 (correct prediction) → algorithm is going in the right direction
   b. Cost value becomes extremely large as h_θ approaches zero and y=1 (incorrect prediction) → large penalty for the algorithm

In this project the formula used is

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

We are therefore aiming to compute $\min_\theta J(\theta)$, which is done through gradient descent:

$$\theta_{k2} = \theta_{k1} - \alpha\left[\sum_{i=1}^{m} (h_\theta\left(x^{(i)}\right) - y^{(i)})x_k^{(i)}\right]$$

Where alpha represents a step size either tuned for the problem or automatically picked.

Note that the superscripts in these equations represent feature indices, not exponents.

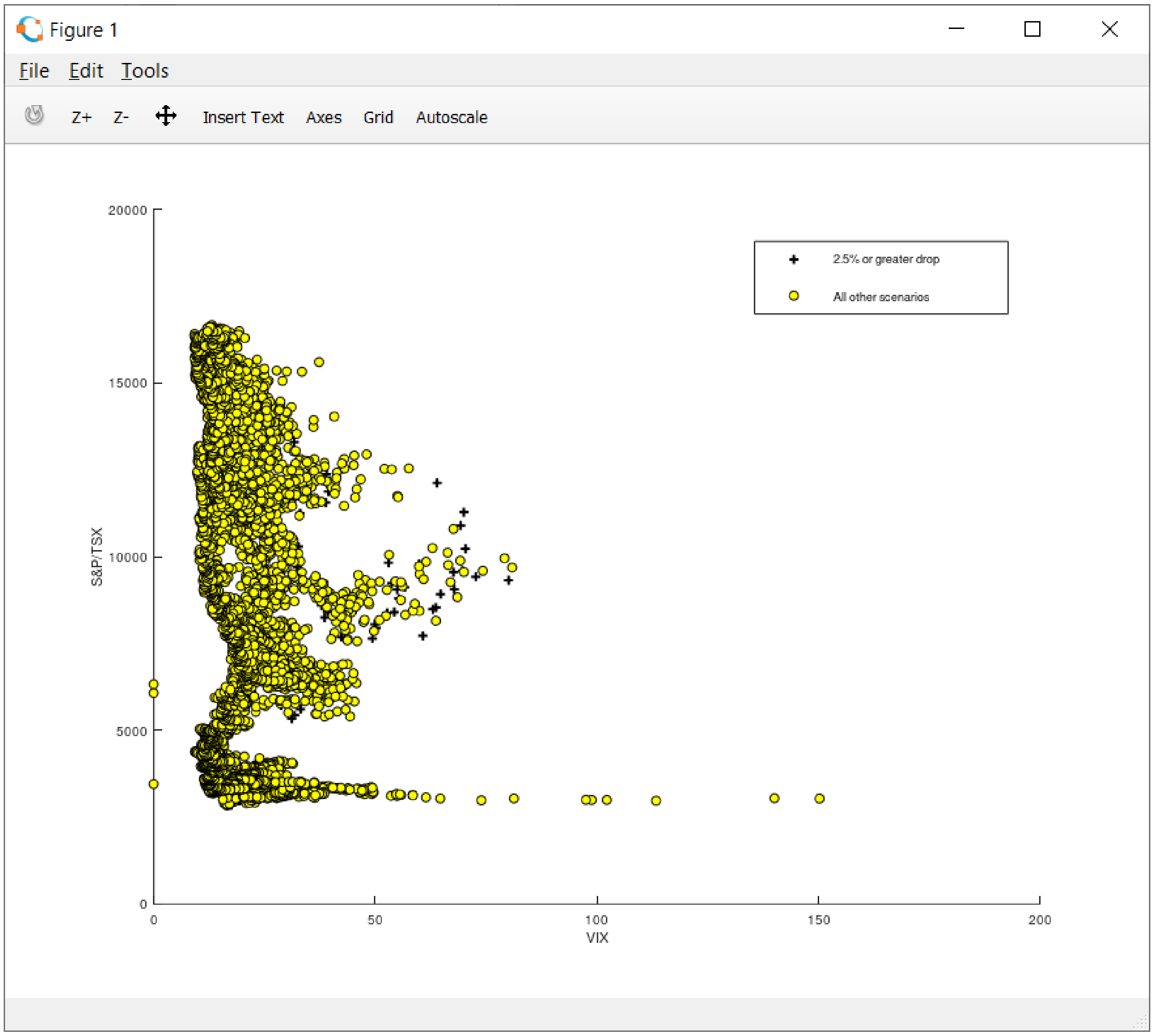
This is iterated with simultaneous updating until the process converges, i.e. the descent per iteration is smaller than an epsilon value, say 10^{-6}.

Extensions with transformed features and multi-class (n>2) classification are also possible with this methodology. Source: https://www.coursera.org/learn/machine-learning

Performing Supervised on Selected VIX Data

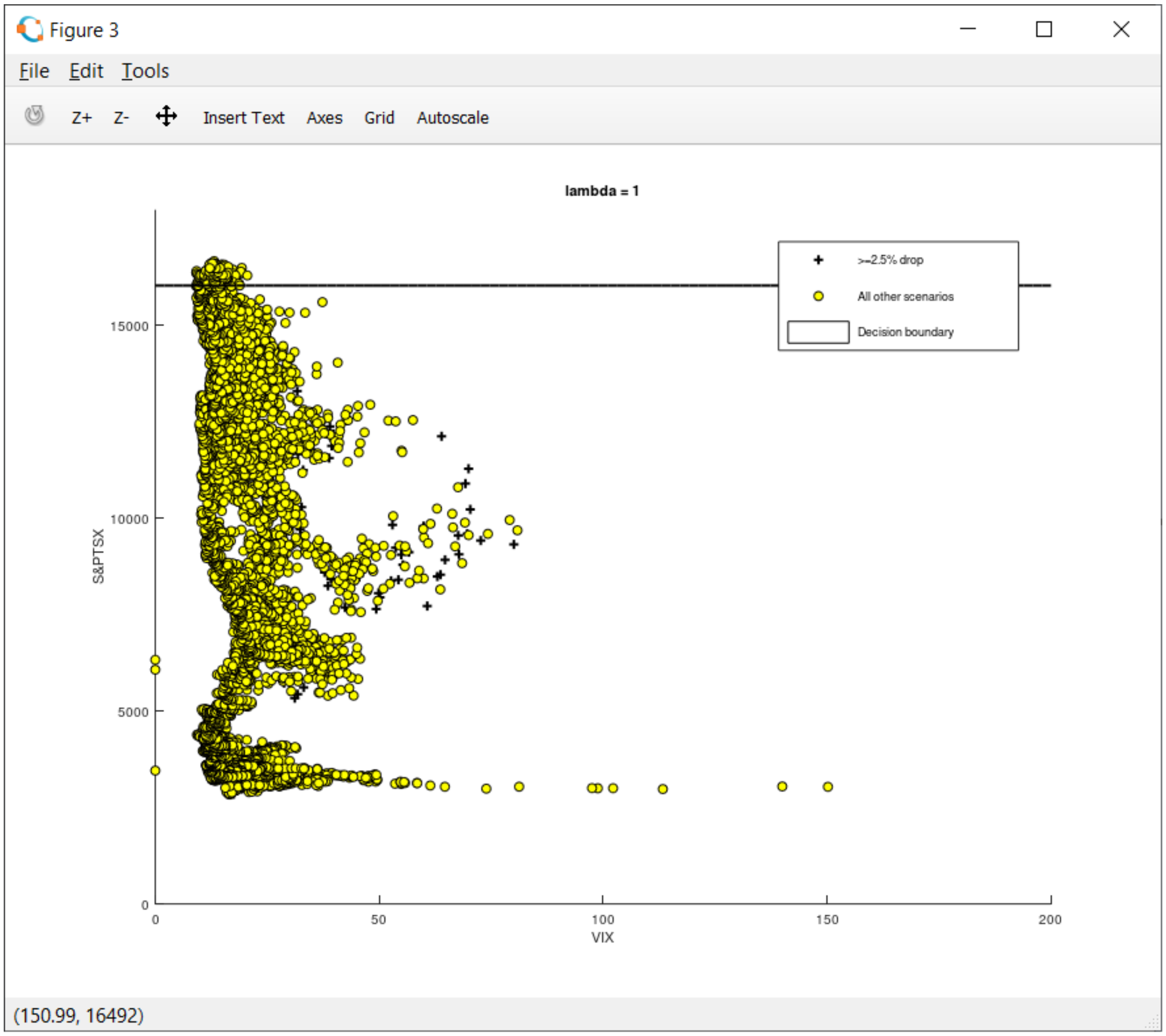
Daily closing values of the VIX and S&P/TSX indices from June 30th, 1986 to July 26th, 2019 were labelled and employed as training data for the binary classifier.

An initial visualization produced the following plot:



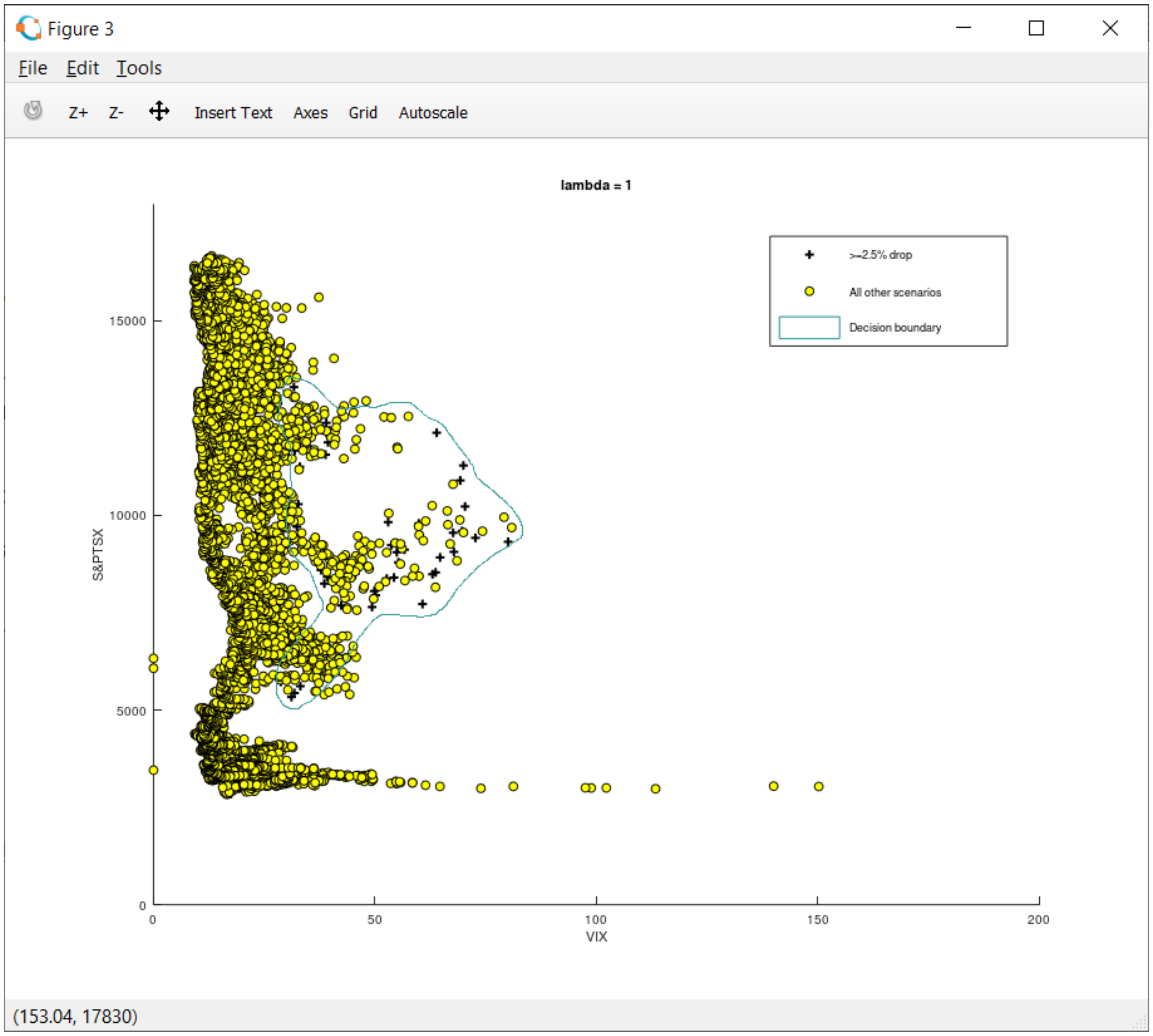
A pattern is immediately visible: the points of interest seem to cluster 'above' the main cloud relative to the vertical axis and are mostly in the approximate range of 50-100 in terms of implied volatility.

Upon performing LR with linear features, a straight line classification is produced:



Clearly this is ineffective as the boundary line does not properly separate the relevant values (y=1).

Adding in polynomial features vastly improves the fitted decision boundary:



With more features plus improved parametrization through regularization (modifying a lambda term to prevent overfitting), this model can certainly be improved to generalize to higher-order nonlinear datasets.