CSCI 181 / E-181 Spring 2014 Practical 3

Kaggle Team "Capt. Jinglehiemer"

David Wihl
davidwihl@gmail.com

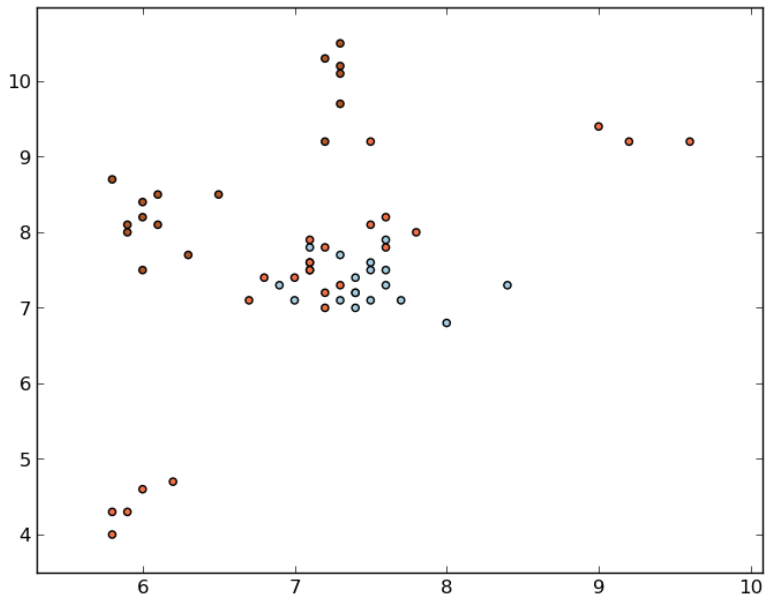
Zack Hendlin
zgh@mit.edu

March 12, 2014

Warm-Up

We consider two approaches for classifying fruits (with length and width measurements provided) into one of three categories.

It is important to note that the data here is not perfectly linearly seperable, as can be seen in the plot below:



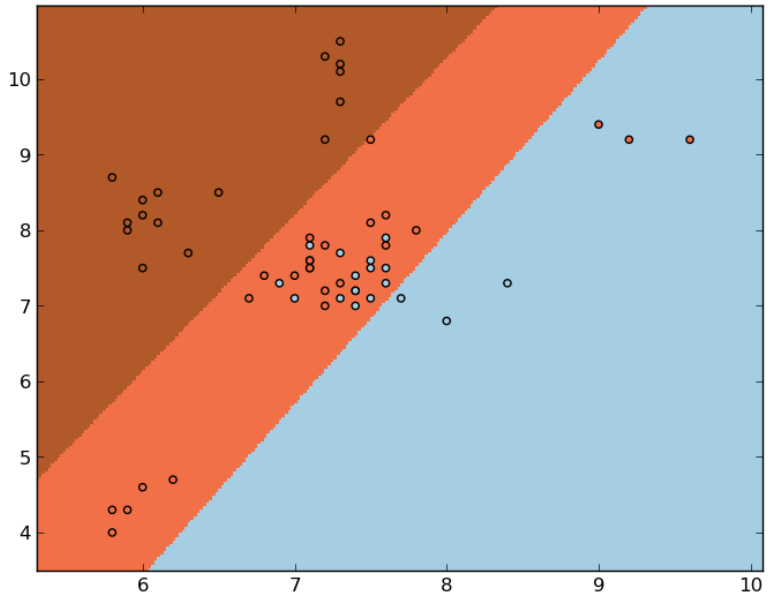
So the challenge becomes how we can best define mutually exclusive regions of the graph.

Logistic Classification

Multiclass logistic classification seeks to define the weights that minimize:
$E(w_1, w_2, ..., w_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} = t_{nk} ln_{nk}$, as given in equation 4.108 in Bishop.
Here our w vector will have be $kx3$ as we have a variable for height, weight, and then an offset term w_0.
Since $k = 3$, we fit 9 weights so as to minimize the error function. We use the Broyden Fletcher Goldfarb Shanno algorithm (BFGS) as implemented in the Scipy package. We select this because it provides a better approximation to the Hessian at each step.
The error function achieves its minimum at 34.769598, and takes 44 iterations to converge. Once we have the weights:

$$w = \begin{bmatrix} -7.35269067 & 2.48983592 & -1.17947522 \\ -3.65199071 & 1.66456348 & -0.81582258 \\ 14.00478476 & -4.15564072 & 1.99410538 \end{bmatrix}$$

we then use in the softmax function: $y_k(\phi) = exp(a_k)/\sum_{j=1}^{K} exp(a_j)$
where a is simply the dot product of x and w,to evaluate which class has the highest likelihood for a particular data point. The point is then assigned to the class which has the highest class-conditional density.
We get the resulting classification.



Generative Model Classification

Generative models attempt to determine, for each class k, a likelihood that a data point is generated by that particular class.
We first calculate the prior likelihoods for each class $P(c_k)$
And then recognizing we want $p(C_k|x_n)$ by applying Bayes rule, we need: $p(x_n|C_k)$.
The multivariate normal is given by:

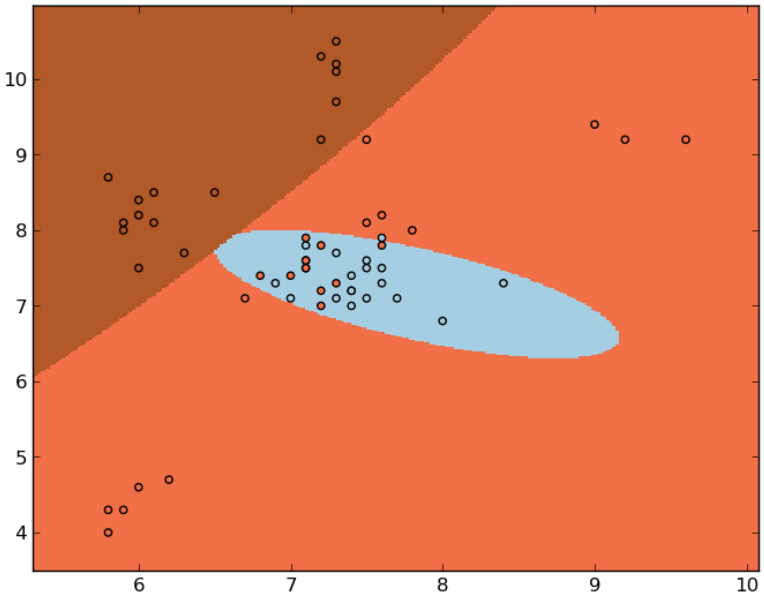
$$f_{\mathbf{x}}(x_1, \ldots, x_k) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right),$$

where k=2 in our case because we have (1) height and (2) width.
This gives us a probability density function for each class. We then apply Bayes rule:

$$p(C_k|x_n) = p(x_n|C_k) * p(C_k) / \sum_{j=1}^{K} p(x_n|c_j) * p(c_j)$$

and note that we can ignore the denominator since it is the same for all classes.

We then have an assignment for each point to the highest likelihood class as shown in plot below.



Warmup Summary

As we can see from the graphs in the previos sections, allowing a non-linear decision boundry (as we did when we used the generative classifier) allowed more flexibility to capture clusters of points and indeed was more accurate at capturing the variation observed in the data.

Classifying Malicious Software

Preliminary Data Analysis

NOTES: 4GB of XML to parse and process, first step was to split the training and the testing. broke into vectorize, train and test steps, persisting appropriate intermediate data at each step. This also enabled parallelization of test runs over a cluster of machines.

Using Cross-validation

Ran 5 CV sets of train / CV data 70/30, 80/20 and 90/10 for each classifier.

Approaches considered

Feature Engineering

Aggregate Features per training file: selected all process features (e.g. 'startreason', 'terminationreason', 'username', 'executionstatus', 'applicationtype') and summary thread features (num of each type of system call).

used CV to generate Logistic Regression weights. Took mean and std of resulting matrix, then eliminated any features where abs(mean) < 0.001 and std <0.01.

Selection of fitting technique

Tried LogisticRegression and SVM with a number of different C values, none of which made a significant difference.

Exploratory Data Analysis

Conclusion