Jericho McLeod
CSI-777
Chapter 7  Review

Somehow, it never occurred to me to weight by euclidean distances for KNN. Having read the sentence, it immediately jumped into the mental category of “why didn’t I think of that” when I first heard about doing this, unlike most tweaks that have been made to models that I read about. An example of something that did not occur to me is the use of constrained quadratic optimization problems to find support vectors. I’ve taken an optimization class, and even looked at simplex problems that closely resemble support vectors, but using them to separate linearly separable classes never crossed my mind, and even reading about it I had to digest the information to make the connection between the model and the optimization problem.

The reason I’m commenting on this is that there seems to be a wide spectrum of possible adjustments to models, from incredibly simple and intuitive through complex and counterintuitive. This leads to the question of whether we are investing heavily in methodologies that are more complex than needed because no one has thought of missing, but relatively simple, adjustments that could be made.

Moving ahead in the book, support vector machines and neural networks are already relatively familiar concepts. The examples in the book are classic introductions, and the gradient descent problem using the derivative of the sigmoid function is something I have implemented manually in the recent past. I have not used radial basis functions, but that seems interesting and relatively easy to implement. Though the text discusses the radial bias function as a specific network type, I see no reason why this could not be applied algorithmically to a neural network, at least experimentally, to check for a change in accuracy or training times. One problem I had in implementing neural networks initially was weights; I instantiated weight variables statically rather than randomly, which led to a consistent output from the entire hidden network. It seems to me that using a gaussian activation function for input vectors would cause divergence for the hidden layer even if it was initialized with a single value for all weights.

Not that this is necessary; it is trivial to initialize weights randomly, but I am intrigued by combining RBF and a deep neural network.

Regression trees are also familiar, though I have not built one from scratch. One of my favorite things about decision trees is the measure of entropy and information gain; regression trees contain a nice little formula for standard deviation reduction that is reminiscent. More interestingly, in my opinion, was examining the author’s pseudo code for  a regression tree. Using objects, rather than data structures, for branches had not occurred to me. I’m going to stop here, lest I spend the rest of the day implementing a manual decision tree model that utilizes objects with associated parameters rather than a data structure.