Christopher Morgan
CS478 : Brother Christophe
Nov 9, 2013

- **Assume that logistic regression applied to a set of supercomputers produces the following simple model: log odds(solution in less than 5 min) = -14.0 + 0.25 NumCPUs. Answer the following questions (all answers should be in terms of e, where e is the inverse function of log; simplify as needed)**
  - What are the odds that a 64 CPU supercomputer will find a solution in less than 5 min?
    Odds = log_odds(64) = exp(2) = 7.389

  - How much better are the odds for a 80 CPU supercomputer (i.e., give the odds ratio)?
    Odds = log_odds(80) = exp(6) = 403.429
    403.429/7.389 = 54.598
    54.598 times better than a 64 CPU

  - What is the probability that a 46 CPU supercomputer will find a solution in less than 5 min?
    P46 = 1/(1+ exp(-(-14 + (0.25 * 64)))) = 0.88

  - What size supercomputer would you need to have a probability 0.73 (=e/(1+e)) of finding a solution in less than 5 minutes?
    CPU = (ln(.73/(1-.73)) + 14)/.25 = 59.9 ~= 60

- **Consider the following transformation (from 3D to 10D space): $f(x) = f(x_1, x_2, x_3) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)$**
  - Show that $K(x, y) = < f(x), f(y) >$ is a kernel function by reducing $K(x, y)$ to $(1 + < x, y >)^2$ (recall that $< x, y >$ is the inner product, i.e., the sum of products of pairwise coordinates).

    $<(1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3), (1, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_3, y_1^2, y_2^2, y_3^2, \sqrt{2}y_1y_2, \sqrt{2}y_1y_3, \sqrt{2}y_2y_3)>$
    $= 1+ 2x_1y_1+ 2x_2y_2+2x_3y_3+x_1^2y_1^2+ x_2^2y_2^2 + x_3^2y_3^2 + x_2^2y_2^2+ 2x_1x_2y_1y_2+2x_1x_3y_1y_3 + 2x_2x_3y_2y_3$
    $= 1+ (2x_1y_1+ 2x_2y_2+2x_3y_3) + (x_1^2y_1^2+ x_2^2y_2^2 + x_3^2y_3^2 + x_2^2y_2^2+ 2x_1x_2y_1y_2+2x_1x_3y_1y_3 + 2x_2x_3y_2y_3)$
    $= 1+ (2x_1y_1+ 2x_2y_2+2x_3y_3) + (x_1y_1+ x_2y_2+x_3y_3) (x_1y_1+ x_2y_2+x_3y_3)$
    $= 1+ 2(x_1y_1+ x_2y_2+x_3y_3) + <x,y> <x,y>$
    $= 1+ 2<x,y> + <x,y> <x,y>$
    $= (1 + <x,y>)^2$

  - Using the SMO implementation of support vector machines in Weka, with the polynomial kernel, verify whether the above transformation is useful in learning the sqrt dataset. You should make sure that the polynomial kernel's degree in SMO is set to 2 (the default is 1) and that the useLowerOrder parameter is set to True. Compare SMO with, for example, VotedPerceptron, which is a simple linear model

learner. Show the values of accuracy for both and discuss your findings. You may want to visualize the data in Weka.

SMO	VotedPerceptron
Exponent: 2 -> Accuracy: 61% Exponent 5 -> Accuracy: 91%	Iterations: 1, Exponent: 1 -> Accuracy: 50% Iterations: 100, Exponent: 1 -> Accuracy: 63% Iterations: 1, Exponent: 2 -> Accuracy: 50% Iterations: 100, Exponent: 2 -> Accuracy: 94%

From the visualizations I learned that by combining both x,y to predict the data's class there appears to be some exponential function that easily separates the data. Using SMO it appears that with the desired exponent the algorithm can achieve ~61% accuracy, however, with 100 iterations and with an exponent equal to 1 VotedPerceptron appears to do just about as well in terms of accuracy. However, I learned that with this particular data set if the exponent is increased in both algorithms the accuracy of both algorithms improves. I have no idea why, save that the exponent deals directly with the line the algorithms are trying to fit to the data, and that a higher exponential function fits the data better. It also appears that VotedPerceptron when given enough iterations produces a slightly more accurate model than the SMO algorithm on this data set.

However, just following the directions literally, it appears that SMO produces a more accurate model than VotedPerceptron with accuracies of 61% and 50% respectively.