MTH 9899: Machine Learning
Bo Feng
Spring 2015

Problem Set # 2
Solution

1. Problem 1

   By definition, the maximum R^2 can be written as

   $$maxR^2 = 1 - \frac{\sum_i (Y_i^* - Y_i)^2}{\sum_i (Y_i^* - \bar{Y_i^*})^2},$$

   where $Y_i^* = Y_i + \varepsilon_i$ is the corrupted signal and Y_i is the original signal. Note that as the number of data points goes to infinity, we have

   $$\begin{aligned} maxR^2 &= 1 - \frac{\sum_i (Y_i^* - Y_i)^2}{\sum_i (Y_i^* - \bar{Y_i^*})^2} \\ &= 1 - \frac{\sum_i \varepsilon_i^2}{\sum_i (Y_i^* - \bar{Y_i^*})^2} \\ &\to 1 - \frac{E(\varepsilon_i^2)}{\text{var}(Y_i^*)} \\ &= 1 - \frac{n^2 \sigma_Y^2}{(n^2+1)\sigma_Y^2} \\ &= 1 - \frac{n^2}{1+n^2}. \end{aligned}$$

   The proof is complete.

2. Problem 2

   Since our approximation of the original signal Y_i is $\hat{Y}_i$ and the observed signal is Y_i^*, the R^2 for our model is therefore

   $$R^2 = 1 - \frac{\sum_i (Y_i^* - \hat{Y}_i)^2}{\sum_i (Y_i^* - \bar{Y_i^*})^2}.$$

   By comparison with Problem 1, as long as we have

   $$\sum_i (Y_i^* - \hat{Y}_i)^2 < \sum_i (Y_i^* - Y_i)^2 = \sum_i \varepsilon_i^2,$$

   the R^2 for our model would appear higher than $maxR^2$. This can be easily achieved by overfitting. (For example, suppose we have N data points. And in the extreme case, we can let our approximation $\hat{Y}_i$ have N parameters, or N degrees of freedom. It will result in $\sum_i (Y_i^* - \hat{Y}_i)^2 = 0$, but it's clearly an over-fitting.)

3. Problem 3

   The risk we face when trading the security stems from the discrepancy between our approximation $\hat{Y_i}$ and the actual return Y_i^*. So the risk can be characterized as $std(Y_i^* - \hat{Y_i})$, or

$$\begin{aligned} std(Y_i^* - \hat{Y_i}) &\approx N^{-1}\sqrt{\sum_i (Y_i^* - \hat{Y_i})^2} \\ &= N^{-1}\sqrt{(1-R^2)\sum_i (Y_i^* - \bar{Y_i^*})^2} \\ &\approx \sqrt{1-R^2}std(Y_i^*), \end{aligned}$$

   where $std(\cdot)$ represents standard deviation, and N is the number of data points. Since R^2 is very close to $maxR^2$, our risk is simply the standard deviation of the actual return scaled by $\sqrt{1-maxR^2}$. If we overestimate R^2, we will underestimate our risk, which will possibly have negative consequences.

4. Problem 4(See the last page)

5. Problem 5

   This is related to the bias-variance trade-off of a model. With more training epochs, the complexity of the model increases. Such adaptation to more complicated underlying structures would often result in decrease in bias. However, this complexity inevitably increases estimation errors, i.e., the variance of the model. Since the validation error is a combination of bias and variance, it will increase at a certain point as a result of the increase in variance, even if the training error continues to decrease.

   The reason that we stop the training when the Validation Error starts to increase is that we are trying to avoid over-fitting problem.

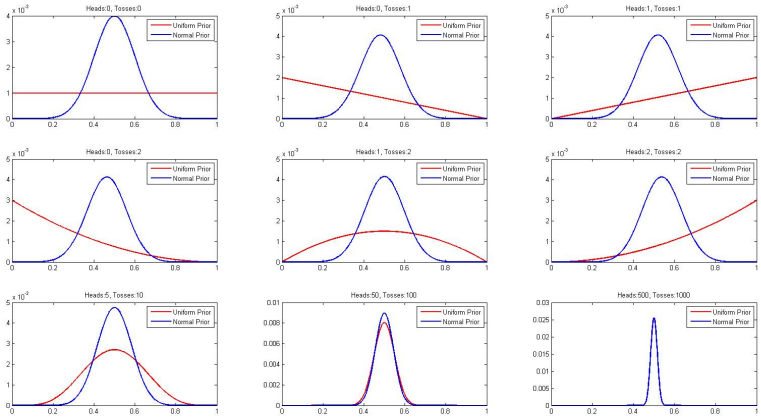
6. Problem 6

   a) The posterior probability is

$$\begin{aligned} P(H|D,I) &\propto P(D|H,I) \times P(H|I) \\ &\propto H^R(1-H)^{N-R} \times \exp\left(-\frac{(H-0.5)^2}{2\times 0.1^2}\right), \end{aligned}$$

   subject to a normalizing constant.

   b)



c) Seen from the plot, the normal prior is more in line with the posterior distribution, and therefore converges to the true value of H more quickly. But as the number of observations increases, the prior has little influence on the posterior distribution. Therefore it wouldn't matter much if we used different priors.

7. Problem 7

   The intuition is that the Occam factor penalizes models that have to be finely tuned to fit the data (in this case, the dispersion is small), favoring models for which the required precision of the parameters is coarse.

8. Problem 8

   This can be used to re-train a new network with a smaller architecture that matches the effective number of degrees of freedom found, or to check if a larger network leads to the same number of degrees of freedom, meaning its unnecessary to go larger.

9. Problem 9

   As shown in the Boltzmann probability, a decrease in temperature parameter will decrease the probability of accepting uphill moves, causing the number of rejections to increase and the step lengths to decline. As the algorithm iterates and the reduction in temperature occurs, acceptance of uphill moves and step lengths decrease. Such monotone temperature schedules allow simulated annealing to explore large features of the cost surface at high T , then perform finer optimization at lower T.

10. Problem 10

    To check the sensitivity of each of the 13 attributes, we can take derivatives of the Median House Value with respect to each attribute. Or, we can perturb each attribute a little bit (say, 1%) and see how much the Median House Value would change accordingly. Generally a larger derivative (compared to the scale of the attribute) would imply the target is sensitive to the attribute.

    It would be a good idea to check collinearities in the 13 attributes, otherwise we would possibly obtain two similar nodes. Suppose the data from the 13 attributes form the design matrix $\mathbf{X}$, we can check whether the smallest eigenvalue of $\mathbf{X}'\mathbf{X}$ is close to zero (or check its condition number). If it is, collinearities exist. We could orthogonalize these 13 attributes (e.g., GramSchmidt orthogonalization) to remove such collinearity.

11. Problem 11

    I think I would get faster learning convergence using two outputs (0,1) and (1,0) to represent classes A and B, then a single output 0 and 1. The higher dimensional output allows for more constraints in the training process, thus the learning converges faster. For example, there are red points and blue points in the 3-D space, and different colored points lies on different but parallel plane. Then 2-D output cannot separate the results, but 3-D output can easily separate the points. Thus higher dimensional outputs can help improve the training process.

    The reason that a thermometer scale produces faster learning convergence is that the thermometer scale has the minimum description length.

12. Problem 12

    It is possible to change the order of preference of the classifiers if we introduce different costs. For example, suppose the cost function is a simple weighted sum of Type I and Type II error,

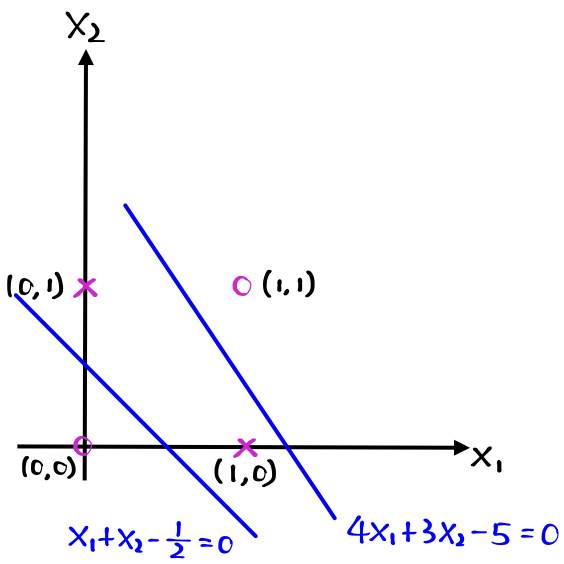
    $$Cost = (1-p) * TypeI + p * TypeII.$$

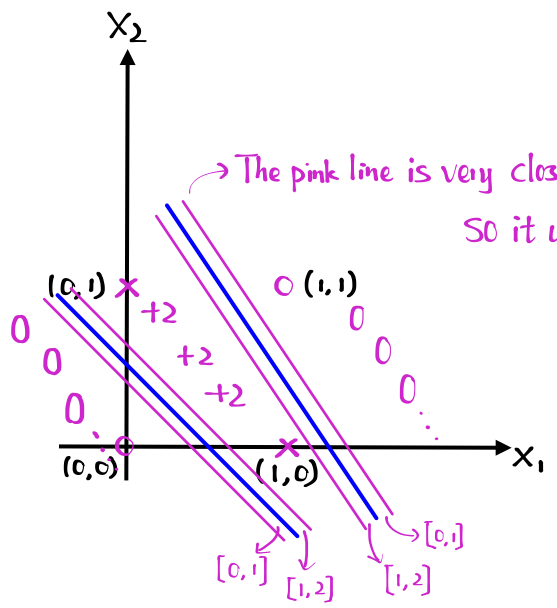
    Letting $p = 0.3$, we have

    $$\begin{aligned} Cost(A) &= 0.3 * 20\% + 0.7 * 5\% = 9.5\%, \\ Cost(B) &= 0.3 * 30\% + 0.7 * 20\% = 23\%, \\ Cost(C) &= 0.3 * 20\% + 0.7 * 30\% = 27\%. \end{aligned}$$

    Since we penalize more on Type II error, clearly B is more preferable than C.

Q4.





Input:

X_1

X_2

$\text{satlin}\left(100(X_1+X_2-\frac{1}{2})\right)$

$\text{satlin}\left(100(5-4X_1-3X_2)\right)$

$$\text{satlin}\left(100(X_1+X_2-\tfrac{1}{2})\right) + \text{satlin}\left(100(5-4X_1-3X_2)\right) = \begin{cases} 1 & \geq 1 \\ 0 & < 1 \end{cases}$$

This is just an example showing that we COULD use satlins for classification.

* Here I use 100 so that as long as $X_1+X_2<\frac{1}{2}$, $\text{satlin}\left(100(X_1+X_2-\frac{1}{2})\right)$ quickly becomes (-1).
There is a very narrow range where it is in $(-1,0]$ and $[0,1)$.