417 hw3

alex.huang

March 2020

1

1.1

By definition, we now have $E_{aug}(w(t)) = E_{in}(w(t)) + \lambda w(t)^T w(t)$. Taking derivative of both sides, we will have $\nabla E_{aug}(w(t)) = \nabla E_{in}(w(t)) + 2\lambda w(t)$. Therefore, we can turn the update rule $w(t+1) \leftarrow w(t) - \eta \nabla E_{aug}(w(t))$ into $w(t+1) \leftarrow w(t) - \eta(\nabla E_{in}(w(t)) + 2\lambda w(t)) = (1 - 2\eta\lambda)w(t) - \eta \nabla E_{in}(w(t))$. That proves the problem.

1.2

If we define a function $sign(x)$, with its value being 1 when $x > 0$, being -1 when $x < 0$, and being 0 when $x = 0$, we can see that $\frac{\partial}{\partial w_i}\|w\|_1 = sign(w_i)$. Therefore, we see the derivative of $\|w\|_1$ being $sign(w)$. Taking derivative on both sides of $E_{aug}(w(t)) = E_{in}(w(t)) + \lambda\|w\|_1$, we will have $\nabla E_{aug}(w(t)) = \nabla E_{in}(w(t)) + \lambda sign(w)$. Rewriting the update rule similar to part 1, we will have $w(t+1) \leftarrow w(t) - \eta \nabla E_{in}(w(t)) - \eta\lambda sign(w(t))$.

1.3

For the case of L2:

When $\lambda = 0.001$, the classification error on test data is 0.1172. There are no 0 on my learned weight vector.

When $\lambda = 0.01$, the classification error on test data is 0.1241. There are no 0 on my learned weight vector.

When $\lambda = 0.05$, the classification error on test data is 0.1241. There are no 0 on my learned weight vector.

When $\lambda = 0.1$, the classification error on test data is 0.1172. There are no 0 on my learned weight vector.

For the case of L1:

When $\lambda = 0.001$, the classification error on test data is 0.1241. There are no 0s on my learned weight vector.

When $\lambda = 0.01$, the classification error on test data is 0.1241. There are no 0s on my learned weight vector, but have one -0.0000.

When $\lambda = 0.05$, the classification error on test data is 0.1241. There are one 0 on my learned weight vector.

When $\lambda = 0.1$, the classification error on test data is 0.1655. There are three 0s on my learned weight vector.

For L2, if we increase lambda, then classification error on test data first increase, then stayed the same, then decreased. However, the changes are insignificant. We can conclude that changing the lambda in this range does not significantly affect the out of sample error.

However, for L1 we see that as lambda grows, the classification error first stayed the same, then drastically improves. Also, the number of 0 increases, which also suggest a strong amount of regularization. The reason 0 increases is probably because with stronger regularization there are bigger chances that $w_i(t+1)$ and $w_i'(t+1)$ have different signs. Therefore, we can say that as lambda grows, we will have a weight of more zero, and there will be stronger regularization. We can also see that for L1, as lambda increases to 0.1 the classification error increases, suggesting that we might have too much regularization, resulting in under-fitting.

Comparing L1, L2, we will conclude that L1 with truncated gradient can sometimes be stronger (or more rigid) regularizer than L2, because it includes more 0s. When lambda is relatively small, L1 and L2 has a similar estimation of out of sample error. However, as lambda=0.1 we see that L1 goes to too much regularization and L2 still has a good classification error.

2

2.1

If we make Γ to be the identity matrix with the dimension same as the number of elements in w, we will have $w^T\Gamma^T\Gamma w = w^T w = \sum_{q=0}^{Q} w_q^2$. Therefore, restraining $w^T\Gamma^T\Gamma w \leq C$ will get us $\sum_{q=0}^{Q} w_q^2 \leq C$.

2.2

If we make Γ to be the row vector with length as the number of elements in w, and we will get $w^T\Gamma^T\Gamma w = (\sum_{q=0}^{Q} w_q)\Gamma w = (\sum_{q=0}^{Q} w_q)(\sum_{q=0}^{Q} w_q) = (\sum_{q=0}^{Q} w_q)^2$. Therefore, restraining $w^T\Gamma^T\Gamma w \leq C$ will get us $(\sum_{q=0}^{Q} w_q)^2 \leq C$.

3

3.1

We should not choose the one with lowest E_{val}. By the VC bound, for the validation result, we have $E_{out} \leq E_{val} + O(\sqrt{\frac{\ln M}{2K}})$. Therefore, if learner A has a smaller E_{val}, and B has a larger K, we cannot simple conclude that the first

one is better, because although A has a smaller first term, B might has a smaller second term, and might eventually has a tighter bound on E_{out}.

3.2

Because the learners all faithfully did the validation using the same data set, we know that their K values are the same. Therefore, we can see that whoever has the smallest E_{val} has the smallest $E_{val} + O(\sqrt{\frac{\ln M}{2K}})$, since the second part is the same for everyone. Therefore, the one with smallest E_{val} has a lowest bound on E_{out} and should be selected.

3.3

Because we're given $k(\epsilon) = -\frac{1}{2\epsilon^2} \ln(\frac{1}{M} \sum_{m=1}^{M} e^{-2\epsilon^2 K_m})$, we have $Me^{-2\epsilon^2 k(\epsilon)} = \sum_{m=1}^{M} e^{-2\epsilon^2 K_m}$, so we only need to prove $P[E_{out}(m*) > E_{val}(m*) + \epsilon] \leq \sum_{m=1}^{M} e^{-2\epsilon^2 K_m}$.

For any m between $1, M$, by Hoeffding's inequality we get $P[E_{out}(m) > E_{val}(m) + \epsilon] \leq e^{-2\epsilon^2 K_m}$. If we put the union bound on $m*$, we will get

$$P[E_{out}(m*) > E_{val}(m*)+\epsilon] \leq P[E_{out}(1) > E_{val}(1)+\epsilon]+\ldots+P[E_{out}(M) > E_{val}(M)+\epsilon] \leq \sum_{m=1}^{M} e^{-2\epsilon^2 K_m}$$

. That proves the problem.

4

4.1

The index SP 500 picked the biggest 500 companies in the market, which constitutes a sampling bias. The reason for that is because bigger companies are more likely to be those that was profitable in the last 50 years. Therefore, the group of 500 companies as a whole is going to be more profitable compared to average companies traded. On the other hand, if we try to replace SP 500 with 500 companies randomly selected, we will be more likely to get the right answer.

This mistake can also be explained as data snooping, because we have chose the 500 companies by looking at the data and selecting the biggest, rather than by random.

If we want to estimate whether the stock is profitable or not, we need to pick the right number of M. Because we have selected the most profitable company that is currently trading and inside SP 500, it is necessary that we use the M=50000 for the Hoeffding bound. The reason for this is that we have consciously or unconsciously searched into the set of all the companies traded to pick for the most profitable one. We cannot use the set of 10000 current trading

companies, because companies not bankrupt also implies it is more likely to be profitable. Therefore, we get the Hoeffding bound of

$$P[|E_{in} - E_{out}| > 0.02] \leq 2 \times 50000 \times e^{-2\times 12500\times 0.02^2} \approx 4.5 > 1$$

Therefore, no conclusion can be generated about whether the stock is profitable.

4.2

We assume that whether stock is profitable is randomly drawn from an unknown distribution that associates with how good the stock is. Therefore, there could be two reasons that SP 500 went up more than half. It could be that some stocks are inherently good, and has a good distribution that brings profitability. The stock could also become profitable by pure luck, because profitability is randomly drawn. The stocks of SP 500 is more likely to select stocks with either a good distribution, a good "luck" historically, or both. If we wish to select any traded stock and "buy and hold", we have no guarantee that it has a good distribution (since we only have profitability data for SP 500), and so the conclusion ("buy and hold" is a good strategy) is wrong. If we try to "buy and hold" with only SP 500 companies, although historically they are profitable we also have no guarantee that the companies have good distribution because SP can also select companies that becomes profitable by luck. In conclusion, the 51 percent result says nothing about the future possibility of buy and hold trading.

There is nothing we can say about buy and hold trading. Any data we get from SP 500 is fundamentally flawed because of sampling bias and data snooping, and therefore cannot be used to draw any conclusion on general buy and hold trading.

5

5.1

We first create the root node, and then (because obviously we do not meet with terminal conditions) choose a root attribution–color, stripes or texture. We use information entropy. Without splitting, we get 60 percent NO and 40 percent YES, so the information entropy is $0.6 \log_2 \frac{1}{0.6} + 0.4 \log_2 \frac{1}{0.4} = 0.97$.

If we split by color, we get the information entropy of red to be 0, because it only has one possibility. That of purple is 1, because half is yes and half is no, and $0.5 \log_2 2 + 0.5 \log_2 2 = 1$. Also, purple has size of 4, and red has size 1. From the formula we get $Gain(D, A) = H(D) - \sum_i \frac{|D_i|}{|D|} H(D_i) = 0.97 - (0\frac{1}{5} + 1\frac{4}{5}) = 0.17$

If we split by stripes, we get the information entropy of no to be 0, because it only has one possibility (no). That of yes is $\frac{1}{3} \log_2 3 + \frac{2}{3} \log_2 \frac{3}{2} = 0.92$, since we have two yes and one no. Also, no has size of 2, and yes has size 3. From the formula we get $Gain(D, A) = H(D) - \sum_i \frac{|D_i|}{|D|} H(D_i) = 0.97 - (0\frac{2}{5} + 0.92\frac{3}{5}) = 0.418$

If we split by texture, we get the information entropy of rough to be 1, because it is half yes and half no, and $0.5\log_2 2 + 0.5\log_2 2 = 1$. That of smooth is $\frac{1}{3}\log_2 3 + \frac{2}{3}\log_2 \frac{3}{2} = 0.92$, since we have two no and one yes. Also, rough has size of 2, and smooth has size 3. From the formula we get $Gain(D, A) = H(D) - \sum_i \frac{|D_i|}{|D|} H(D_i) = 0.97 - (1\frac{2}{5} + 0.92\frac{3}{5}) = 0.018$

Therefore, we choose the choice that maximizes my Gain(D,A), which is to split by stripes. Therefore, the root attribute of the tree is whether the mushroom has stripes.

5.2

Procedure:

If no stripes, only have one outcome, predict NO.

If has stripes, we can see that obviously color has more information gain (color perfectly splits results), therefore split by color.

If color is red, predict NO.

If purple, predict YES.

A complete tree is shown below: