1: AML Pg. 17, Ex. 1.7

Answer:

a) $\mathcal{H}$ has two hypotheses, one always with $\circ$ and one always with $\bullet$. The $\circ$ hypothesis is chosen if there are more $\circ$ labels in the training data $\mathcal{D}$, and $\bullet$ is chosen if there are more $\bullet$ labels. In the five training points given, three are $\bullet$ and two are $\circ$, so the $\bullet$ hypothesis is chosen. Each of the three test points outside of $\mathcal{D}$, therefore, are labeled $\bullet$ as per the hypothesis g.

Of the 8 possible target functions f_1, ..., f_8, only f_8 labels all three points $\bullet$. Two possible target functions, f_7 and f_4 agree with g on exactly two of the points. Three possible target functions – f_2, f_3, and f_5 – agree with g on exactly one of the points. f_1 agrees with g on none of the points.

b) If the learning algorithm now picks the hypothesis that matches the data set the least, it will choose a hypothesis that always returns $\circ$, as $\circ$ is the minority outcome in the training data. One possible target function, f_1, agrees with g on all three points being $\circ$. Two possible target functions, f_2 and f_5, agree with g on two points being $\circ$. Three possible target functions – f_4, f_6, and f_7 – agree with g on exactly one of the points. f_8 agrees with g on none of the points.

c) The hypothesis that returns $\bullet$ if the number of 1's in the point $\mathbf{x}$ is odd, and $\circ$ otherwise, assigns the following classifications to the test data: 101 – $\circ$; 110 – $\circ$; and 111 – $\bullet$. Only f_2 agrees with this g on all three points. Three possible target functions – f_1, f_4, and f_6 – agree with g on exactly two points. Three possible target functions – f_3, f_5, and f_8 – agree with g on exactly one point. f_7 agrees with g on none of the points.

d) To agree completely with the training data, but disagree the most with the XOR of part c, the hypothesis must assign $\bullet$ if $\mathbf{x}$ has exactly one one, and $\circ$ otherwise. It is clear that this produces the correct classifications for the training data. For the test data, this hypothesis classifies each point as $\circ$, as each has more than one one. This hypothesis g, therefore, only agrees with the XOR hypothesis on 2 of 3 test points. This hypothesis yields the same results as the hypothesis of part b: One possible target function, f_1, agrees with g on all three points being $\circ$. Two possible target functions, f_2 and f_5, agree with g on two points being $\circ$. Three possible target functions – f_4, f_6, and f_7 – agree with g on exactly one of the points. f_8 agrees with g on none of the points.

2: AML Pg. 19, Ex. 1.8

Answer:

The probability of picking a red marble is $\mu = 0.9$. For a sample of 10 marbles, the probability that 0 or 1 will be red ($\nu \leq 0.1$) can be found using the binomial distribution:

$$P(k; n, \mu) = \binom{n}{k} \mu^k (1-\mu)^{(n-k)}$$

where k is the number of red marbles in the sample and n is the number of marbles in the sample. The probability that 0 *or* 1 marble will be red is the sum of probabilities:

$$P(\nu \leq 0.1; n = 10, \mu = 0.9) = P(k = 1; n = 10, \mu = 0.9) + P(k = 0; n = 10, \mu = 0.9)$$

where $\nu = \frac{k}{n}$ is the fraction of red marbles in the sample.

$$P(\nu \leq 0.1; n = 10, \mu = 0.9) = \binom{10}{1} 0.9^1 (1 - 0.9)^{(10-1)} + \binom{10}{0} 0.9^0 (1 - 0.9)^{(10-0)}$$

$$P(\nu \leq 0.1; n = 10, \mu = 0.9) = \frac{10!}{9!\,1!} 0.9(0.1)^9 + \frac{10!}{10!\,0!} 1(0.1)^{10}$$

$$P(\nu \leq 0.1; n = 10, \mu = 0.9) = 10 * 0.9 * 10^{-9} + 1 * 10^{-10}$$

$$P(\nu \leq 0.1; n = 10, \mu = 0.9) = 0.9 * 10^{-8} + \frac{1}{100} * 10^{-8}$$

$$P(\nu \leq 0.1; n = 10, \mu = 0.9) = 0.91 * 10^{-8}$$

$$P(\nu \leq 0.1; n = 10, \mu = 0.9) = 9.1 * 10^{-9}$$

3: AML Pg. 19, Ex. 1.9

Answer:

Hoeffding Inequality:

$$\mathbb{P}[|\nu - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N}$$

If the population probability is $\mu = 0.9$ and the sample size is $N = 10$ marbles, then the probability of achieving a sample fraction $\nu \leq 0.1$ is bounded by summing the Hoeffding bounds for $\nu = 0.1$ and $\nu = 0$. The summation arises because $\nu \leq 0.1$ is satisfied if either $\nu = 0.1$ or $\nu = 0$.

$$\mathbb{P}[|0.1 - 0.9| > \epsilon] \leq 2e^{-2\epsilon^2 * 10}$$
$$\mathbb{P}[0.8 > \epsilon] \leq 2e^{-20\epsilon^2}$$

$$\mathbb{P}[|0 - 0.9| > \epsilon] \leq 2e^{-2\epsilon^2 * 10}$$
$$\mathbb{P}[0.9 > \epsilon] \leq 2e^{-20\epsilon^2}$$

$\therefore$

$$P(\nu \leq 0.1, \epsilon) = \mathbb{P}[0.8 > \epsilon] + \mathbb{P}[0.9 > \epsilon] \leq 4e^{-20\epsilon^2}$$

Say $\epsilon = 0.1$:

$$P(\nu \leq 0.1, \epsilon = 0.1) \leq 4e^{-20*(0.1)^2}$$

$$P(\nu \leq 0.1, \epsilon = 0.1) \leq 3.275$$

Say $\epsilon = 1.0$:

$$P(\nu \leq 0.1, \epsilon = 1.0) \leq 4e^{-20*(1.0)^2}$$
$$P(\nu \leq 0.1, \epsilon = 1.0) \leq 8.245 * 10^{-9}$$

As we impose a higher lower bound (ϵ) on the difference between population and sample probabilities, the probability of achieving a difference greater than that bound decreases. This is because a higher difference between population and sample probabilities reflects increasing dissimilarity between the population and sample, which is unlikely. For a lower bound ϵ around 1.0, therefore, the probability of achieving our desired sample difference (reflecting $\nu \leq 0.1$) approaches the binomial distribution probability of $\nu \leq 0.1$ as given in Ex. 1.8, to the order of 10^{-9}.

4: AML Pg. 25, Ex. 1.11

Answer:

25 training examples from unknown target function. Hypotheses $\mathcal{H}$: $h_1 = +1$, $h_2 = -1$. Learning algorithm S chooses the hypothesis that most agrees with training examples. Learning algorithm C chooses the other hypothesis.

a) From the deterministic perspective, $\mathcal{D}$ cannot tell us anything about data outside of $\mathcal{D}$, so a hypothesis h_S that agrees the most with training examples (produced by S) is *not* guaranteed to agree the most with a majority of test examples. That is, such a hypothesis is not guaranteed to perform better than random. Likewise, a hypothesis h_C that disagrees the most with training examples is not guaranteed to perform worse than random.

From the probabilistic perspective, however, the Hoeffding inequality says that the difference between in- and out-of-sample error rates is only greater than an arbitrary threshold ϵ with a determined probability ($= 2Me^{-2\epsilon^2 N}$). Given 25 data points, the probability of E_{out} differing from E_{in} by ϵ decreases as ϵ increases. In effect, this probability is a guarantee that, on average, a hypothesis h_S agreeing with the sample (training) data will achieve lower E_out than h_C (disagreeing with the sample data) will. This is because of the bounded probability on the difference between E_{out} and E_{in}.

b) If all examples in $\mathcal{D}$ have $y_n = +1$, from the deterministic persepective, this does not imply anything about the success of a hypothesis produced from either C or S. From the probabilistic perspective, the hypothesis produced by S will result in $E_{in} = 0$. By the Hoeffding inequality, the probability of E_{out} being different from this increases as the difference increases; in other words, h_S is likely to achieve near-zero out-of-sample error. At the very least, E_{out} of h_S is more likely to be closer to zero than the E_{out} of h_C. Again, this is due to a combination of the Hoeffding Inequality and the fact that $E_{in} = 1$ for h_C (every point is misclassified).

c) If $p = 0.9$, $\mathbb{P}[f(x) = +1] = 0.9$. That is, the probability of having a label equal to +1 in the population is 0.9. On average, then, 90% of training examples will equal +1, so learning algorithm S will choose h_1, achieving $E_{in} = 0.1$ on average, and learning algorithm C will choose h_2, achieving $E_{in} = 0.9$ on average. Given $p = 0.9$ for the entire population, on average $E_{out} = 0.1$

for h_1 and $E_{out} = 0.9$ for h_2. The probability of S producing a better hypothesis than C is therefore given as follows:

$$\mathbb{P}[E_{out}(h_1) < E_{out}(h_2)] = \mathbb{P}[0.1 < 0.9] = 1$$

d) No, there is not a value of p for which it is more likely than not that C will produce a better hypothesis than C. In part c, I showed that $p = 0.9$ leads to a better outcome from S. Indeed, this is the case for all $p > 0.5$. Take, for example, $p = 0.51$. In that case, $h_S = h_1$ and $h_C = h_2$. From this, we get $E_{out}(h_1) = 0.49$ and $E_{out}(h_2) = 0.51$. $\mathbb{P}[0.49 < 0.51] = 1$

For the case of $p = 0.5$, $h_S = h_1$ or $h_S = h_2$ with equal probability, depending on the sample. The same goes for h_C. In any case, $E_{out}(h_S) = 0.50$ and $E_{out}(h_C) = 0.50$, so C does not produce a better hypothesis than S.

For the case of $p < 0.5$, $h_S = h_2$ and $h_C = h_1$ since more points are -1 than +1. For example, say $p = 0.3$. In this case, h_2 correctly classifies 70% of points in the population, and h_1 correctly classifies 30% of points in the population. Therefore, $E_{out}(h_2) = 0.3$ and $E_{out}(h_1) = 0.7$, so

$$\mathbb{P}[E_{out}(h_S) < E_{out}(h_C)] = \mathbb{P}[E_{out}(h_2) < E_{out}(h_1)] = \mathbb{P}[0.3 < 0.7] = 1$$

meaning S produces a better hypothesis than C.

5: AML Pg. 26, Ex. 1.12

Answer:

If the 4000 data points are generated independently, and if the out-of-sample data on which the friend intends to use g are generated by the same distribution as the 4000 data points, then I will provide her with a g that has a high probability of approximating f well out of sample. If either of these conditions is not met, any g I produce will not have a high probability of approximating f well out of sample. The answer is therefore c.

6: AML Pg. 31, Ex. 1.13

Answer:

$$P(y|x) = \begin{cases} \lambda & y = f(x) \\ 1 - \lambda & y \neq f(x) \end{cases}$$

a) In the deterministic case, h makes an error with probability μ in approximating f. In the noisy case, f is given by $P(y|x)$ instead of $y = f(x)$. This $P(y|x)$ is λ if $y = f(x)$.

The probability of error that h makes in approximating y is therefore $1 - (1 - \mu)\lambda$, as h first has a probability $1 - \mu$ of approximating $P(y|x)$ correctly, then has a probability λ of that $P(y|x)$ corresponding to $y = f(x)$, so the probability of h correctly approximating $y = f(x)$ is $(1 - \mu)\lambda$.

b) The performance of h will be independent of μ if $\lambda = 0$, because in that case the noise is such that $y = f(x)$ does not occur for any x. This is reflected in the probability of h approximating y correctly: $1 - (1 - \mu) * 0 = 1$

7: AML Pg. 92, Ex. 3.6

Answer:

a) The maximum likelihood error can be written, as per Pg. 91 of AML, as follows:

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln \frac{1}{\theta(y_n \mathbf{w}^T \mathbf{x}_n)}$$

Note that the hypothesis is defined as $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x}_n)$. If $y_n = 1$, therefore, the argument of the logarithm becomes $\frac{1}{\theta(\mathbf{w}^T\mathbf{x}_n)} = \frac{1}{h(\mathbf{x}_n)}$.

Likewise, $y_n = -1$, the argument of the logarithm becomes $\frac{1}{\theta(-\mathbf{w}^T\mathbf{x}_n)} = \frac{1}{1-\theta(\mathbf{w}^T\mathbf{x}_n)} = \frac{1}{1-h(\mathbf{x}_n)}$ by the property (AML Pg. 91) $1 - \theta(s) = \theta(-s)$.

If we define $[\![y_n = +1]\!]$ to equal 1 when the statement inside the double brackets is true, and zero otherwise, then $[\![y_n = +1]\!] \ln \frac{1}{h(\mathbf{x}_n)}$ equals $\ln \frac{1}{h(\mathbf{x}_n)}$ when $y_n = +1$ and zero otherwise.

Likewise, $[\![y_n = -1]\!] \ln \frac{1}{1-h(\mathbf{x}_n)}$ equals $\ln \frac{1}{1-h(\mathbf{x}_n)}$ when $y_n = -1$ and zero otherwise.

We can therefore rewrite the error function as follows:

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} [\![y_n = +1]\!] \ln \frac{1}{h(\mathbf{x}_n)} + [\![y_n = -1]\!] \ln \frac{1}{1 - h(\mathbf{x}_n)}$$

since, as shown above, the argument of the log in either term is equivalent to $\frac{1}{\theta(y_n \mathbf{w}^T \mathbf{x}_n)}$ if the statement within the term's double brackets is true, and zero otherwise.

b) If $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$, the result of part a can be written as follows:

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} [\![y_n = +1]\!] \ln \frac{1}{\theta(\mathbf{w}^T\mathbf{x}_n)} + [\![y_n = -1]\!] \ln \frac{1}{1 - \theta(\mathbf{w}^T\mathbf{x}_n)}$$

Minimizing this error is equivalent to setting its derivative w.r.t. the weights equal to zero:

$$\frac{d}{d\mathbf{w}} E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} [\![y_n = +1]\!] \frac{d}{d\mathbf{w}} \ln \frac{1}{\theta(\mathbf{w}^T\mathbf{x}_n)} + [\![y_n = -1]\!] \frac{d}{d\mathbf{w}} \ln \frac{1}{1 - \theta(\mathbf{w}^T\mathbf{x}_n)}$$

where the derivative can be moved within the sum over points because the weights are independent of any given point. By the chain rule and the fact that $\frac{d}{dx} \ln x = \frac{1}{x}$:

$$\frac{d}{d\mathbf{w}} E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} [\![y_n = +1]\!] \theta(\mathbf{w}^T\mathbf{x}_n) \frac{d}{d\mathbf{w}} \frac{1}{\theta(\mathbf{w}^T\mathbf{x}_n)} + [\![y_n = -1]\!] (1 - \theta(\mathbf{w}^T\mathbf{x}_n)) \frac{d}{d\mathbf{w}} \frac{1}{1 - \theta(\mathbf{w}^T\mathbf{x}_n)}$$

$$\frac{d}{d\mathbf{w}} E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} [\![y_n = +1]\!] \theta(\mathbf{w}^T\mathbf{x}_n) \frac{d}{d\mathbf{w}} (\theta(\mathbf{w}^T\mathbf{x}_n))^{-1} + [\![y_n = -1]\!] (1 - \theta(\mathbf{w}^T\mathbf{x}_n)) \frac{d}{d\mathbf{w}} (1 - \theta(\mathbf{w}^T\mathbf{x}_n))^{-1}$$

Again, by the chain rule:

$$\frac{d}{d\mathbf{w}}E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}[\![y_n = +1]\!]\theta(\mathbf{w}^T\mathbf{x}_n)[-(\theta(\mathbf{w}^T\mathbf{x}_n))^{-2}]\frac{d}{d\mathbf{w}}\theta(\mathbf{w}^T\mathbf{x}_n)$$

$$+[\![y_n = -1]\!](1-\theta(\mathbf{w}^T\mathbf{x}_n))[-(1-\theta(\mathbf{w}^T\mathbf{x}_n))^{-2}]\frac{d}{d\mathbf{w}}(1-\theta(\mathbf{w}^T\mathbf{x}_n))$$

Simplifying...

$$\frac{d}{d\mathbf{w}}E_{in}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}[\![y_n = +1]\!]\frac{\frac{d}{d\mathbf{w}}\theta(\mathbf{w}^T\mathbf{x}_n)}{\theta(\mathbf{w}^T\mathbf{x}_n)} + [\![y_n = -1]\!]\frac{\frac{d}{d\mathbf{w}}(1-\theta(\mathbf{w}^T\mathbf{x}_n))}{1-\theta(\mathbf{w}^T\mathbf{x}_n)}$$

The derivative of the logistic function is given as follows (see proof in previous homeworks):

$$\frac{d}{dx}\theta(x) = \theta(x)(1-\theta(x))$$

$\therefore$ and applying the chain rule:

$$\frac{d}{d\mathbf{w}}E_{in}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}[\![y_n = +1]\!]\frac{\theta(\mathbf{w}^T\mathbf{x}_n)(1-\theta(\mathbf{w}^T\mathbf{x}_n))\mathbf{x}_n}{\theta(\mathbf{w}^T\mathbf{x}_n)} + [\![y_n = -1]\!]\frac{\theta(\mathbf{w}^T\mathbf{x}_n)(1-\theta(\mathbf{w}^T\mathbf{x}_n))\mathbf{x}_n}{1-\theta(\mathbf{w}^T\mathbf{x}_n)}$$

$$\frac{d}{d\mathbf{w}}E_{in}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}[\![y_n = +1]\!](1-\theta(\mathbf{w}^T\mathbf{x}_n))\mathbf{x}_n + [\![y_n = -1]\!]\theta(\mathbf{w}^T\mathbf{x}_n)\mathbf{x}_n$$

Recalling that $1-\theta(s) = \theta(-s)$...

$$\frac{d}{d\mathbf{w}}E_{in}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}[\![y_n = +1]\!]\theta(-\mathbf{w}^T\mathbf{x}_n)\mathbf{x}_n + [\![y_n = -1]\!]\theta(\mathbf{w}^T\mathbf{x}_n)\mathbf{x}_n$$

In a similar but reversed technique from part a, the argument of the sum can be rewritten to produce the following:

$$\frac{d}{d\mathbf{w}}E_{in}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\theta(-y_n\mathbf{w}^T\mathbf{x}_n)\mathbf{x}_n$$

because if $y_n = +1$, the argument of the logistic function becomes $-\mathbf{w}^T\mathbf{x}_n$ as expected, and if if $y_n = -1$, the argument of the logistic function becomes $\mathbf{w}^T\mathbf{x}_n$ as expected. Substituting in the logistic function...

$$\frac{d}{d\mathbf{w}}E_{in}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\frac{e^{-y_n\mathbf{w}^T\mathbf{x}_n}}{1+e^{-y_n\mathbf{w}^T\mathbf{x}_n}}\mathbf{x}_n$$

Equation 3.9:

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln(1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n})$$

Again, minimizing the error is equivalent to setting its derivative equal to zero:

$$\frac{d}{d\mathbf{w}} E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \frac{d}{d\mathbf{w}} \ln(1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n})$$

By the chain rule and the fact that $\frac{d}{dx} \ln x = \frac{1}{x}$:

$$\frac{d}{d\mathbf{w}} E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n}} \frac{d}{d\mathbf{w}} (1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n})$$

By the chain rule and the fact that $\frac{d}{dx} e^x = e^x$:

$$\frac{d}{d\mathbf{w}} E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n}} e^{-y_n \mathbf{w}^T \mathbf{x}_n} (-y_n \mathbf{x}_n)$$

Simplifying...

$$\frac{d}{d\mathbf{w}} E_{in}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N} \frac{e^{-y_n \mathbf{w}^T \mathbf{x}_n}}{1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n}} y_n \mathbf{x}_n$$

The derivatives of the result of part a and of Eq. 3.9 are equivalent, so setting each to zero results in the same minimization for both.

8: AML Pg. 98, Ex. 3.10

Answer:

a) Define the point error as follows:

$$e_n(\mathbf{w}) = \max(0, -y_n \mathbf{w}^T \mathbf{x}_n)$$

PLA first identifies a misclassified point. SGD, by contrast, randomly selects a point and determines its error. However, if SGD picks a correctly classified point, $y_n \mathbf{w}^T \mathbf{x}_n > 0$, meaning $-y_n \mathbf{w}^T \mathbf{x}_n < 0$ so the point error as defined above is $e_n(\mathbf{w}) = 0$. Any derivative of zero is zero, so no weight update will occur. SGD, then, only performs updates for misclassified points, like PLA.

PLA next updates the weights based on the error of a misclassified point, as follows (AML Pg. 7): $\Delta \mathbf{w}_n = y_n \mathbf{x}_n$. Note that the book uses t to denote the number of weight updates so far, where t corresponds to a specific point n. I use n to reinforce the PLA-SGD analogy.

For SGD, the weight update is the following: $\Delta \mathbf{w}_n = \eta \mathbf{v}_n$. The direction vector $\mathbf{v}_n$ is defined as the negative of the gradient of the error (point error given above):

$$\mathbf{v}_n = -\nabla_{\mathbf{w}} e_n(\mathbf{w}) = \nabla_{\mathbf{w}} y_n \mathbf{w}^T \mathbf{x}_n = y_n \mathbf{x}_n$$

$$\therefore \Delta \mathbf{w}_n = \eta y_n \mathbf{x}_n = y_n \mathbf{x}_n$$

if $\eta = 1$. The weight updates of PLA and SGD are therefore the same.

b) In the case of logistic regression with SGD, $\nabla e_n(\mathbf{w}) = \frac{-y_n \mathbf{x}_n}{1+e^{y_n \mathbf{w}^T \mathbf{x}_n}}$. As the weights grow, the second term either grows or shrinks exponentially, depending on if the point is classified correctly.

Say the weights are large. If the point is classified correctly, the exponent of the denominator's second term is positive and large, so the denominator is large and the gradient as a whole is small. The weights are hardly adjusted for a correctly classified point. This is similar to PLA, which does not adjust weights at all for a correctly classified point.

If the point is misclassified, the exponent of the denominator's second term is negative and has a large magnitude. The second term of the denominator is therefore close to zero, so the gradient is about equal to the numerator: $\nabla e_n(\mathbf{w}) \approx -y_n \mathbf{x_n}$. The weight update is therefore $\Delta \mathbf{w}_n = -\eta \nabla e_n(\mathbf{w}) \approx \eta y_n \mathbf{x_n}$. If $\eta = 1$, this is almost equivalent to a PLA weight update.

9: AML Pg. 123, Ex. 4.2

Answer:

For each scenario (Q_f, N, σ), the out-of-sample test MSE tends to be lower for g_2 than for g_{10}. We can observe this by looking at elements of equal index in the "o2AvgTestMSEs" and "o10AvgTestMSEs" lists calculated in the Jupyter notebook for this problem. Elements of the former tend to be smaller than elements of the latter, suggesting that a degree-10 polynomial overfits most Legendre polynomial target functions with added noise σ, for most numbers of data points. Indeed, only four configurations result in an average out-of-sample MSE lower for the degree-10 polynomial – indexes 17, 73, 92, and 119 of the AvgTestMSEs lists. Tellingly, indexes 17, 73, and 119 correspond to the 15-Legendre-polynomials target function, and index 92 corresponds to the 10-Legendre-polynomials target function. Overfitting occurs less (10th order polynomials perform better) when the target function is more complex.

10: PRML Pg. 285, Ex. 5.9

Answer:

Output $-1 \leq y(\mathbf{x}, \mathbf{w}) \leq 1$ and target values $t = 1$ for C_1 and $t = -1$ for C_2.

Say we have an output unit activation function $y(\mathbf{x}, \mathbf{w})$, which is the probability of the input $\mathbf{x}$ belonging to C_1 ($t = 1$): $y(\mathbf{x}, \mathbf{w}) = p(C_1|\mathbf{x})$. The probability of the input belonging to the other class C_2 ($t = -1$), then, is $1 - y(\mathbf{x}, \mathbf{w}) = p(C_2|\mathbf{x})$. The sigmoid function can assign values between zero and one. We want a function that can assign values between negative one and one. To achieve this, first scale the sigmoid by a factor of two such that it can assign values between zero and two. Then, shift the sigmoid by negative one such that it assigns values between negative one and one. Put mathematically:

$$y(a) = 2\sigma(a) - 1$$

For a sigmoid function, the conditional distribution of targets was given as follows, where the "s" subscript stands for sigmoid:

$$p(t|\mathbf{x}, \mathbf{w}) = y_s(\mathbf{x}, \mathbf{w})^{t_s}(1 - y_s(\mathbf{x}, \mathbf{w}))^{1-t_s}$$

yielding the negative log likelihood error function of PRML Eq. 5.21:

$$E(\mathbf{w}) = -\sum_{n=1}^{N}[t_{s,n} \ln y_{s,n} + (1 - t_{s,n}) \ln(1 - y_{s,n})]$$

In the above error function, $y_{s,n} = \sigma(\mathbf{w}^T\mathbf{x})$. This can be written in terms of our new activation function $y_n = 2\sigma(\mathbf{w}^T\mathbf{x}) - 1$, as follows:

$$y_{s,n} = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{y_n + 1}{2}$$

The sigmoid target $t_{s,n}$ can be written in terms of the new target in the same way:

$$t_{s,n} = \frac{t_n + 1}{2}$$

We can now rewrite PRML Eq. 5.21 in terms of the new target:

$$E(\mathbf{w}) = -\sum_{n=1}^{N}[\frac{t_n + 1}{2} \ln \frac{y_n + 1}{2} + (1 - \frac{t_n + 1}{2}) \ln(1 - \frac{y_n + 1}{2})]$$

Simplifying the second term of the sum...

$$E(\mathbf{w}) = -\sum_{n=1}^{N}[\frac{t_n + 1}{2} \ln \frac{y_n + 1}{2} + (\frac{2}{2} - \frac{t_n + 1}{2}) \ln(\frac{2}{2} - \frac{y_n + 1}{2})]$$

$$E(\mathbf{w}) = -\sum_{n=1}^{N}[\frac{t_n + 1}{2} \ln \frac{y_n + 1}{2} + \frac{1 - t_n}{2} \ln \frac{1 - y_n}{2}]$$

We can use the fact that the log of a quotient can be written as the log of the dividend minus the log of the divisor to extract a log of two from both terms:

$$E(\mathbf{w}) = -\sum_{n=1}^{N}[\frac{t_n + 1}{2} \ln(y_n + 1) + \frac{1 - t_n}{2} \ln(1 - y_n) - 2 \ln 2]$$

Pulling the factor of $\frac{1}{2}$ out of the sum and acknowledging that $-2 \ln 2$ summed N times is $-2N \ln 2$...

$$E(\mathbf{w}) = -\frac{1}{2}\sum_{n=1}^{N}[(t_n + 1) \ln(y_n + 1) + (1 - t_n) \ln(1 - y_n)] - 2N \ln 2$$

We can drop this $-2N \ln 2$ because it is independent of the weights and therefore does not affect error minimization.

$$E(\mathbf{w}) = -\frac{1}{2}\sum_{n=1}^{N}[(t_n + 1)\ln(y_n + 1) + (1 - t_n)\ln(1 - y_n)]$$

The above is the error function for targets of -1 and 1 and outputs in-between and including -1 and 1. The activation function, written explicitly, is as follows:

$$\begin{aligned} y(a) = 2\sigma(a) - 1 &= \frac{2}{1 + e^{-a}} - 1 \\ &= \frac{2}{1 + e^{-a}} - \frac{1 + e^{-a}}{1 + e^{-a}} \\ &= \frac{1 - e^{-a}}{1 + e^{-a}} \\ &= \frac{1 - e^{-a}}{1 + e^{-a}} * \frac{e^{a/2}}{e^{a/2}} \\ &= \frac{e^{a/2} - e^{-a/2}}{e^{a/2} + e^{-a/2}} \\ &= \tanh(a/2) \end{aligned}$$

by the definition of the hyperbolic tangent function.

11: PRML Pg. 289, Ex. 5.28

Answer: In a non-convolutional network, the derivative of the error for an example n with respect to a weight w_{ji} (between nodes j and i) is given by PRML Eq. 5.50:

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}}$$

where w_{ji} is the weight between nodes j and i, each in a different layer, and a_j is the activation of node j.

Weight-sharing means that backpropagation of errors must be modified in convolutional layers. Specifically, the errors of all units in a feature map m of a convolutional layer contribute to the feature map's error gradient. Summing errors over all units (each labeled j) in the mth feature map, Eq. 5.50 becomes the following:

$$\frac{\partial E_n}{\partial w_i^{(m)}} = \sum_j \frac{\partial E_n}{\partial a_j^{(m)}}\frac{\partial a_j^{(m)}}{\partial w_i^{(m)}}$$

where $w_i^{(m)}$ denotes the ith element of the feature vector used to produce the mth feature map when $w^{(m)}$ convolves with the previous layer. The error derivative is subscripted n to indicate that this is the error derivative for the nth training example.

Note that the jth unit's activation $a_j^{(m)}$ is the product of the input to j (from element i in the previous layer or input) and the weight along that connection, $w_i^{(m)}$, that is: $a_j^{(m)} = w_i^{(m)} z_{ji}^{(m)}$. Taking the derivative of this activation, we get the following:

$$\frac{\partial a_j^{(m)}}{\partial w_i^{(m)}} = z_{ji}^{(m)}$$

If we define the derivative of the error with respect to the activation to be $\delta_j^{(m)} = \frac{\partial E_n}{\partial a_j^{(m)}}$, then the derivative of the error with respect to the weight i becomes the following:

$$\frac{\partial E_n}{\partial w_i^{(m)}} = \sum_j \delta_j^{(m)} z_{ji}^{(m)}$$

Say layer j precedes layer k. Computing δ_k is done using information from layer j, as per PRML Eq. 5.55:

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j}$$

If k is a convolutional layer's feature map, $\frac{\partial E_n}{\partial a_k}$ may seem problematic, as weight-sharing means that this is the same for any value of k. While this is true, it is not an impediment to the algorithm's success, as the sum of Eq. 5.55 is not used in updating the feature vector to layer k. The weights are still updated as a group, and Eq. 5.55 can be used as it is for a non-convolutional network.