cse7/4641 Problem Set 1

Bin Xie

Febrary 2019

Answer 1.1

The data comes from a non-deterministic function, we can have:

$$P(D|h) = \prod_{i=1}^{m} P(x_i, d_i|h)$$

The x is independent of h, we can have:

$$P(D|h) = \prod_{i=1}^{m} P(x_i, d_i|h) = \prod_{i=1}^{m} P(d_i|h, x_i)P(x_i)$$

Here, we have:

$$P(d_i|h, x_i) = \left\{ \begin{array}{ll} h(x_i), & d_i = 1 \\ (1 - h(x_i)), & d_i = 0 \end{array} \right. = h(x_i)^{d_i}(1 - h(x_i))^{1-d_i}$$

We substitute this equation into the above equation, we can have:

$$P(D|h) = \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} P(x_i)$$

For the definition of ML hypothesis, we can have:

$$\begin{aligned} h_{ML} &= \underset{h \in H}{\arg\max} \prod_{i=1}^{m} P(D|h) \\ &= \underset{h \in H}{\arg\max} \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} P(x_i) \end{aligned}$$

The last term $P(x_i)$ is a constant which is independent of h, it can be dropped. So we can have:

$$\begin{aligned} h_{ML} &= \underset{h \in H}{\arg\max} \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} \\ &= \underset{h \in H}{\arg\max} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)) \end{aligned}$$

Answer 1.2

For the rule we derived for a deterministic function perturbed by zero-mean gaussian noise, minimizing the sum of the squared errors produces the maximum likelihood hypothesis. By contrast, for this problem, the rule that maximizes the above equation seeks the maximum likelihood hypothesis.

For a normal neural network that seeks to minimize the sum of squared errors, it can perform a gradient ascent rather than gradient descent search and the weight vector is adjusted in the direction of the gradient, using the following update rule:

$$w_{jk} \leftarrow w_{jk} + \triangle w_{jk}$$

where

$$\triangle\, w_{jk} = \eta sum_{i=1}^{m}(d_i - h(x_i))x_{ijk}$$

$\triangle w_{jk}$ is derived as following: denote the quantity in 1.1 equation as $G(h, D)$

$$\begin{aligned}
\frac{\partial G(h,D)}{\partial w_{jk}} &= \sum_{i=1}^{m} \frac{\partial G(h,D)}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}} \\
&= \sum_{i=1}^{m} \frac{\partial(d_i \ln h(x_i) + (1-d_i)\ln(1-h(x_i))}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}} \\
&= \sum_{i=1}^{m} \frac{d_i - h(x_i)}{h(x_i)(1-h(x_i))} \frac{\partial h(x_i)}{\partial w_{jk}}
\end{aligned}$$

To keep the analysis simple, suppose our neural network is constructed from a single layer of sigmoid units. In this case we have

$$\frac{\partial h(x_i)}{\partial w_{jk}} = \sigma^{'}(x_i)x_{ijk} = h(x_i)(1-h(x_i))x_{ijk}$$

$$\frac{\partial G(h,D)}{\partial w_{jk}} = \sum_{i=1}^{m}(d_i - h(x_i))x_{ijk}$$

If the data consisted of x,y pairs where y was an estimate of the probability instead of 0s and 1s, it will be more like the deterministic function. The most likely hypothesis will be the one that minimize the sum of squared errors between the observed training values yi and the hypothesis prediction $h(x_i)$.

Answer 2

Design a two-input perceptron that implements the boolean function $A \wedge\ \neg B$
The weights are: $w_0 = -0.8, w_1 = 0.5, w_2 = -0.5$

$x_1(A)$	$x_2(B)$	$w_0 + w_1x_1 + w_2x_2$	output
-1	-1	-0.8	-1
-1	1	-1.8	-1
1	-1	0.2	1
1	1	-0.8	-1

Design a two-layer network of perceptrons that implements $A \oplus\ B$
The weights are:
Hidden layer1: $w_0 = -0.8, w_1 = 0.5, w_2 = -0.5$
Hidden layer2: $w_0 = -0.8, w_1 = -0.5, w_2 = 0.5$
Output: $w_0 = 0.3, w_1 = 0.5, w_2 = 0.5$

x$_1(A)$	x$_2(B)$	Hidden layer1	Hidden layer2	output
-1	-1	-1	-1	-1
-1	1	-1	1	1
1	-1	1	-1	1
1	1	-1	-1	-1

Answer 3

The perceptron training rule updates perceptron weights:

$$w_i = w_i + \triangle w_i,\ \triangle w_i = \eta(y - o)x_i$$

Here, w_i is the weight associated with the i th input, η is the learning rate (like .01), y is the current training example's output value, o is the output of the perceptron under the current training example.

The gradient descent training rule specifies how the weights are to be changed at each step of the learning procedure so that the prediction error of the unit decreases the most.

The gradient descent training rule updates perceptron weights:

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \sum_{x\in X}(out_x - o_X)\frac{\partial}{\partial w_i}(out_x - (w_0 + w_1x_{1x} + w_1x_{1x}^2 + ... + w_nx_{nx} + w_nx_{nx}^2)) \\ &= \sum_{x\in X}(out_x - o_X)(-x_{ix} - x_{ix}^2)\end{aligned}$$

The advantage of gradient descent training rule over perceptron training rule is that it can deal with non-linearly separable dataset through a best-fit approximation. However, perceptron training rule will fail to converge.

Answer 4

To use Decision Trees to perform regression, we can use standard deviation reduction as splitting criteria. The whole process is as follows:

Step1: Calculate the standard deviation for the current training dataset.

Step2: For each feature, calculate its standard deviation and the standard deviation reduction.

Step3: Pick the feature with highest standard deviation reduction as the new node in the tree.

Step4: Divide the dataset into subsets according to the values of the feature.

Step5: Run recursively the process on the non-leaf branches until all data is processed. When the number of instances is more than one at a leaf node, the mean is calculated as the final value.

Since the dataset link isn't valid in the problem set 1, so I search for the dataset by myself. The link is https://archive.ics.uci.edu/ml/machine-learning-databases/housing/ .

I use the 10-fold cross validation, and the RMSE is 4.4076 and MAE is 2.9954.

Answer 5

We can still use the information gain as the splitting criterion to build the decision tree just like ID3 algorithm. However, we only build the tree on the branch that has the value of the attribute in the test samples. Therefore, we only construct one path in the decision tree and the other paths which are not relevant to the test samples are ignored.

The advantage is that we can save exponential time by not building the parts of the decision tree that we don't need. Also, if the number of attributes are small, it can perform faster.

The disadvantage is that when the size of test samples increases, we may still end up building the whole tree to classify all sample tests and the computation can be expensive.

Answer 6

For this problem, we have an instance space of points on the plane and we need to find a linear target function like $f(x) = w_0 + w_1 x$.

For decision trees, we will use regression decision tree. To decide on the best splitting, we have to go through infinite pairs of w_0, w_1 to calculate the standard deviation reduction. Besides, we have to take all training samples for function approximation. The computation time is long. However, regression trees will not be hugely affected by noise and outliners.

For nearest neighbor regression, the computation is much less expensive. However, the choice of k which is the number of nearest neighbors has a critical influence on function approximation. If the k is small, the noise can have a huge influence on the regression performance. If k is large, the computation time will increase a lot.

So if we can choose a proper value for k, I will use nearest neighbor learning for this problem. Or if it's hard to find a good value of k and we need to perform a lot of queries, I will use decision tree learning for this problem.

Answer 7.1

The VC dimension of an origin-centered circle (2D) is 2. With any set of three points, they will be at some radii $r1 \leq r2 \leq r3$ from the origin, and no function f will be able to label the points at r1 and r3 with +1 while labeling the point at r2 with 1. Thus, no set of three points is shatterable by origin-centered circles, making the VC dimension just 2.

Answer 7.2

The VC dimension of an origin-centered sphere (3D) is 2. The same reasoning as for the 2D case applies.