6.867: Exercises (Week 8)

October 28, 2016

Contents

[1]Uni. of Toronto STA 414 2014 Practice Problem Set 2, Question 1

[2]Uni. of Toronto STA 414 2014 Practice Problem Set 2, Question 2

[3]CMU 15-381 Fall 2001 Homework 5, Problem 2

[4]CMU 15-381 Fall 2001 Homework 5, Problem 3

Solution: Don't look at the solutions until you have tried your absolute hardest to solve the problems.

1 Need a smoothie?*

Your friend David just learned a new technique called Gaussian process and he's trying to generate some 1D examples to build some intuition on the different covariance functions for a zero-mean Gaussian process prior. Unfortunately he accidentally spilled the smoothie his girlfriend bought him over his laptop and destroyed his laptop that contains the code he used to generate the plots. Conveniently he has printed out some plots for different covariance functions earlier and roughly remembers what kind of functions he used to generate these plots. As a good friend who excelled in 6.867, you are trying to comfort him by labeling the plots.

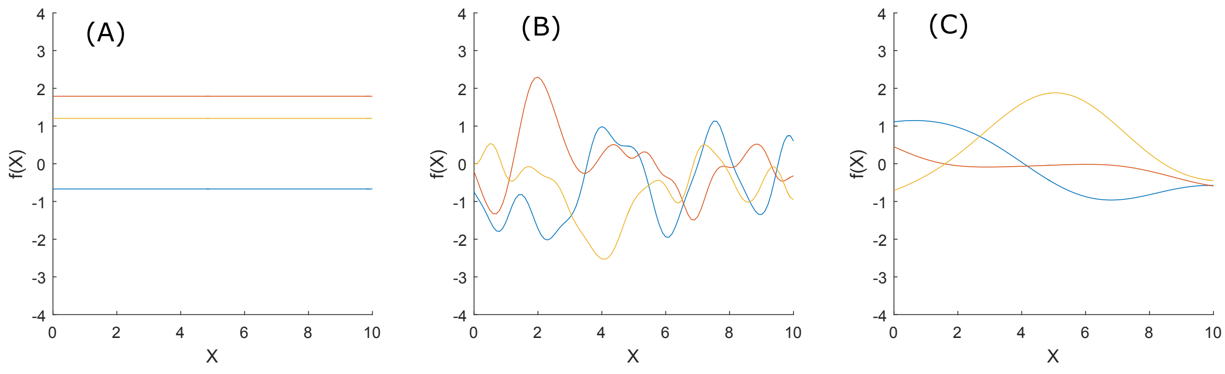
(a) The following plots contain random functions drawn from a covariance function with a squared exponential kernel $k(x, z) = \exp(-\frac{1}{2\tau^2}\|x - z\|^2)$ with different values of τ. Indicate which one of them corresponds to:

i. $\tau = 0.5$

ii. $\tau = 3$

iii. $\tau \to \inf$

Solution: (A): $\tau \to \mathtt{inf}$. (B) $\tau = 0.5$. (C) $\tau = 3$.



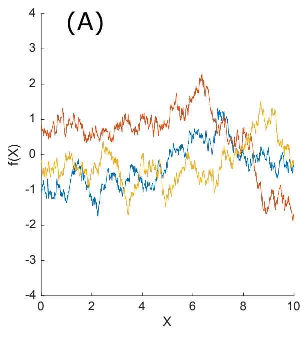
(b) Qualitatively describe what your random function would look like if you draw with a squared exponential kernel when $\tau \to 0$.

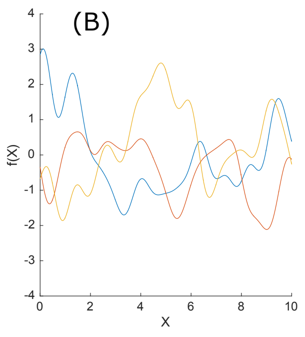
Solution: Function values drawn across the X axis would be marginally independent of each other regardless of how close they are in X.

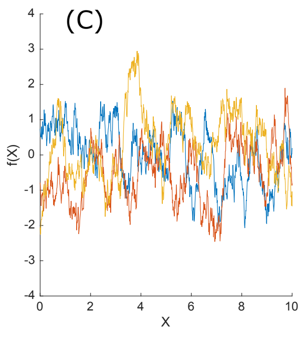
(c) In addition to the squared exponential kernel $k(x, z) = \exp(-\frac{1}{2\tau^2}\|x - z\|^2)$, David has also tried an exponential kernel in the form of $k(x, z) = \exp(-\frac{\|x-z\|}{\tau})$. He has tried two values for

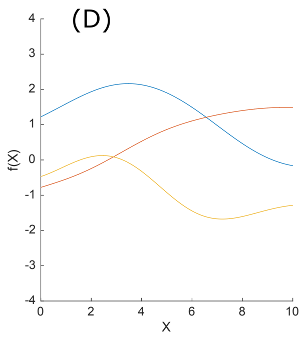
τ for both kernels and the values are $\tau = 3$ and $\tau = 0.5$. For each of the plots below, indicate which kernel function it is generated with which τ value.

Solution: (A): Exponential kernel with $\tau = 3$. (B): Squared exponential kernel with $\tau = 0.5$. (C): Exponential kernel with $\tau = 0.5$. (D): Squared exponential kernel with $\tau = 3$.









2 Calculation with squared exponential kernel*

Consider a Gaussian process with $\mathrm{Mean}[f(x)] = 0$, and a squared exponential kernel of the form

$$k(x, z) = \sigma_f^2 \exp\left(\frac{-||x - z||^2}{2l^2}\right),$$

where l is the characteristic length scale and σ_f is the signal standard deviation. Assume we get observations of y values with noise variance σ_N^2.

Let $l^2 = 0.5$, $\sigma_f^2 = 5$, $\sigma_N^2 = 0.01$. Assume you have been given observations $((1, 1), (2, -1))$ (these are points in (x, y) space). What is the mean and variance of the prediction at a new query point $x_* = 1.5$? Write down the solution in terms matrices and vectors, you don't need to hand-calculate the final numerical results[5]

Solution: For our training data, we can construct the covariance matrix $K(x, x)$ as

$$K(x, x) = \begin{bmatrix} 5 & 5e^{-1} \\ 5e^{-1} & 5 \end{bmatrix}$$

We can also construct the covariance vector for our new query point x_* with respect to the training data

$$K(x_*, x) = [5e^{-0.25}, 5e^{-0.25}]$$

Our posterior predictive mean is

[5] If you are trying to refer to parameter estimate formula from the textbook "Gaussian Processes for Machine Learning", notice that in eq. (2.21), the predictive distribution is derived with respect to f_*, in order to derive the predictive distribution for y_*, you need to add $\sigma_n^2 I$ to the lower right block of the covariance matrix. This would also impact subsequent equations such as eq. (2.24).

$$\mu_* = K(x_*, x)(K(x,x) + \sigma_N^2 I)^{-1} y = [5e^{-0.25}, 5e^{-0.25}] \begin{bmatrix} 5+0.01 & 5e^{-1} \\ 5e^{-1} & 5+0.01 \end{bmatrix}^{-1} [1, -1]^\top = 0$$

Our posterior predictive variance is

$$\begin{aligned} \sigma_*^2 &= K(x_*, x_*) + \sigma_N^2 I - K(x_*, x)(K(x,x) + \sigma_N^2 I)^{-1} K(x, x_*) \\ &= 5 + 0.01 - [5e^{-0.25}, 5e^{-0.25}] \begin{bmatrix} 5+0.01 & 5e^{-1} \\ 5e^{-1} & 5+0.01 \end{bmatrix}^{-1} [5e^{-0.25}, 5e^{-0.25}]^\top = 0.5824 \end{aligned} \tag{2.1}$$

3 Simple kernel [6]

Suppose we model the relationship of a real-valued response variable, y, to a single real input, x, using a Gaussian process model in which the mean is zero and the covariances of the observed responses are given by

$$\mathrm{Cov}(y_i, y_j) = 0.5^2 \delta_{i,j} + K(x_i, x_j)$$

with the noise-free covariance function, K, defined by

$$K(x, x') = \begin{cases} 1 - |x - x'| & |x - x'| < 1 \\ 0 & otherwise \end{cases}$$

Suppose we have four training cases, as follows:

x	y
0.5	2.0
2.8	3.3
1.6	3.0
3.9	2.7

Recall that the conditional mean of the response in a test case with input x_*, given the responses in the training cases, is $k^T C^{-1} y$, where y is the vector of the training responses, C is the covariance matrix of training responses, and k is the vector of covariances of training responses with the response in the test case.

Find the predictive mean for the response in a test case in which the input is $x_* = 1.2$.

Solution: The covariance matrix of the training responses is

$$C = \begin{bmatrix} 1+0.5^2 & 0 & 0 & 0 \\ 0 & 1+0.5^2 & 0 & 0 \\ 0 & 0 & 1+0.5^2 & 0 \\ 0 & 0 & 0 & 1+0.5^2 \end{bmatrix}$$

[6]Uni. of Toronto STA 414 2014 Practice Problem Set 2, Question 1

The inverse of this is

$$C^{-1} = \begin{bmatrix} 0.8 & 0 & 0 & 0 \\ 0 & 0.8 & 0 & 0 \\ 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0.8 \end{bmatrix}$$

The vector of covariances of the test response with the training response is

$$k = \begin{bmatrix} 1-0.7 \\ 0 \\ 1-0.4 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0 \\ 0.6 \\ 0 \end{bmatrix}$$

Therefore $k^T C^{-1} = [0.24, 0, 0.48, 0]$, and the predictive mean for the test response is

$$k^T C^{-1} y = 0.24 \times 2.0 + 0.48 \times 3.0 = 1.92$$

4 Covariance or not? [7]

Recall that for a Gaussian process model the predictive distribution of the response y_* in a test case with inputs x_* has mean and variance given by

$$E[y_*|x_*, training\ data] = k^T C^{-1} y$$
$$Var[y_*|x_*, training\ data] = \nu - k^T C^{-1} k,$$

where y is the vector of observed responses in training cases, C is the matrix of covariances for the responses in training cases, k is the vector of covariances of the response in the test case with the responses in training cases, and ν is the prior (co)variance of the response in the test case.

(a) Suppose we have just one training case, with $x_1 = 3$ and $y_1 = 4$. Suppose also that the noise-free covariance function is $K(x, x') = 2^{-|x-x'|}$, and the variance of the noise is 1/2. Find the mean and variance of the predictive distribution for the response in a test case for which the value of the input is 5.

Solution: The equations given above doesn't take into account of the noise of our measurement, if we add in the noise term σ^2, the mean and variance for a new input x_* would be

$$E[y_*|x_*, training\ data] = k^T (C + \sigma^2)^{-1} y$$
$$Var[y_*|x_*, training\ data] = \nu + \sigma^2 - k^T (C + \sigma^2)^{-1} k$$

Therefore, the mean of the predictive distribution is

$$K(3,5)[K(3,3) + 1/2]^{-1}(4) = (1/4)[1 + 1/2]^{-1}(4) = 4/6$$

[7]Uni. of Toronto STA 414 2014 Practice Problem Set 2, Question 2

The variance of the predictive distribution is

$$[K(5,5)+1/2]-K(3,5)[K(3,3)+1/2]^{-1}K(3,5) = [1+1/2]-(1/4)[1+1/2]^{-1}(1/4) = 35/24$$

(b) Repeat the calculations, but using $K(x,x') = 2^{+|x-x'|}$. What can you conclude from the result of this calculation?

Solution:

The mean of the predictive distribution is

$$K(3,5)[K(3,3)+1/2]^{-1}(4) = (4)[1+1/2]^{-1}(4) = 32/3$$

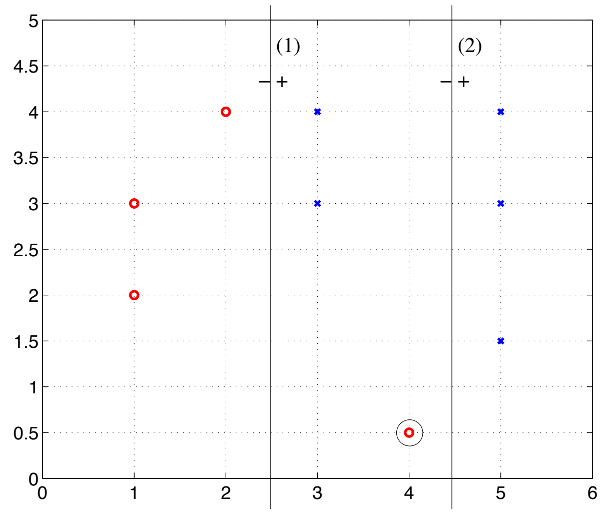
The variance of the predictive distribution is

$$[K(5,5)+1/2]-K(3,5)[K(3,3)+1/2]^{-1}K(3,5) = [1+1/2]-(4)[1+1/2]^{-1}(1/4) = -55/6$$

Notice that the variance in this case is a negative number, which is clearly wrong. We can therefore conclude that $K(x,x') = 2^{+|x-x'|}$ is not a valid covariance function – it is not positive semi-definite.

5 AdaBoost with decision stumps*

Consider the labeled training points below, where x and o denote positive and negative labels, respectively. We wish to apply AdaBoost with decision stumps to solve the classification problem. In each boosting iteration, we select the stump that minimizes the weighted training error, breaking ties arbitrarily.



(a) On the figure above, draw the decision boundary corresponding to the first decision stump that the boosting algorithm would choose. Label this boundary (1), and also indicate +/- side of the decision boundary.

(b) On the same figure also circle the point(s) that have the highest weight after the first boosting iteration.

(c) What is the weighted error of the first decision stump after the first boosting iteration, i.e., after the points have been reweighted?

Solution:
First, calculate the vote α_1

$$\alpha_1 = \log(\frac{1-1/9}{1/9}) = \log 8$$

Then the weight the misclassified point (let it be point k) is updated.

$$w_2^{(k)} = w_1^{(k)} exp^{\log 8} = \frac{8}{9}$$

The weight of all other eight points are kept unchanged (still equal to 1/9).

Hence the weighted error is:

$$E_W = \frac{w_2^{(k)}}{8 \times 1/9 + w_2^{(k)}} = \frac{8/9}{8/9 + 8/9} = 0.5$$

(d) Draw the decision boundary corresponding to the second decision stump, again on the figure, and label it with (2), also indicating the +/- side of the boundary.

(e) Would some of the points be misclassified by the combined classifier after the two boosting iterations? Provide a brief justification.

Solution: Yes. For example, the circled point in the figure is misclassified by the first decision stump and could be classified correctly in the combination only if the weight/votes of the second stump is higher than the first. If it were higher, however, then the points misclassified by the second stump would be misclassified in the combination.

6 Generalized boosting

Note the version of boosting described here is a generalization of the one in the class notes and in Bishop. For exponential loss, we get a version of boosing that is similar to the one we have seen; there are some detailed differences (for example, the weight are normalized) but they are not fundamental.

Boosting is a simple procedure for estimating ensembles

$$h_m(x) = \sum_{j=1}^{m} \alpha_j h(x; \theta_j)$$

where $\alpha_j \geqslant 0$ (not necessarily summing to one). The base learners $h(x; \theta_j)$ are often simple (weak) classifiers such as decision stumps. There are many variations on the boosting algorithm due to the choice of base learners or the loss function that the overall algorithm seeks to minimize.

At stage m of the algorithm, we we will try to minimize

$$J(\alpha_m, \theta_m) = \sum_{i=1}^{n} \text{Loss}\Big(y^i h_m(x^i)\Big) = \sum_{i=1}^{n} \text{Loss}\Big(y^i h_{m-1}(x^i) + y^i \alpha_m h(x^i; \theta_m)\Big)$$

where we assume that $h_{m-1}(x)$ is fixed from $m-1$ previous rounds and view J as a function of parameters α_m and θ_m associated with the new base classifier. The margin loss $\text{Loss}(\cdot)$ could be any function that is convex and decreasing in its argument (smaller loss for predictions with larger voting margin).

Let's elaborate on the algorithm a bit. We can initialize $h_0(x) = 0$ for the ensemble with no base learners. Then, after $m-1$ rounds, we perform two distinct steps to optimize θ_m and α_m, respectively. First, we find parameters $\hat{\theta}_m$ that minimize

$$\frac{\partial}{\partial \alpha_m} J(\alpha_m, \theta_m)\bigg|_{\alpha_m = 0} = \sum_{i=1}^{n} \overbrace{dL\Big(y^i h_{m-1}(x^i)\Big)}^{-W_{m-1}(i)} y^i h(x^i; \theta_m) = -\sum_{i=1}^{n} W_{m-1}(i)\, y^i h(x^i; \theta_m)$$

where $dL(z) = d/dz\, \text{Loss}(z)$ is always negative because the loss is decreasing. (We applied the chain rule of derivatives to get the above form.) The values of the weights will differ based on different loss functions but they will always decrease as a function of how well the current ensemble classifies the training examples. We can also normalize the weights in the algorithm since the normalization does not affect the resulting $\hat{\theta}_m$.

Once we have $\hat{\theta}_m$, we can find $\hat{\alpha}_m$ that minimizes

$$J(\alpha_m, \hat{\theta}_m) = \sum_{i=1}^{n} \text{Loss}\left(y^i h_{m-1}(x^i) + y^i \alpha_m h(x^i; \hat{\theta}_m)\right)$$

We usually will not get a closed-form expression for $\hat{\alpha}_m$. However, the optimization problem is easy to solve since $\text{Loss}(\cdot)$ is convex and we only have one parameter to optimize.

We can now write the general boosting algorithm more succinctly. After initializing $W_0(i) = 1/n$, each boosting iteration consists of the following three steps:

(Step 1) Find $\hat{\theta}_m$ that (approximately) minimizes the weighted error ϵ_m or $2\epsilon_m - 1$ given by

$$-\sum_{i=1}^{n} W_{m-1}(i)\, y^i h(x^i; \theta_m)$$

(Step 2) Find $\hat{\alpha}_m$ that minimizes

$$J(\alpha_m, \hat{\theta}_m) = \sum_{i=1}^{n} \text{Loss}\left(y^i h_{m-1}(x^i) + y^i \alpha_m h(x^i; \hat{\theta}_m)\right)$$

(Step 3) Reweight the examples

$$W_m(i) = -c_m \, dL\left(y^i h_{m-1}(x^i) + y^i \hat{\alpha}_m h(x^i; \hat{\theta}_m)\right)$$

where c_m normalizes the new weights so that they sum to one.

Now that we have a boosting algorithm for any loss function, we can select a particular one. Specifically, we will consider the logistic loss:

$$\text{Loss}(z) = \log\left(1 + \exp(-z)\right)$$

(a) Show that the unnormalized weights $W_m(i)$ from the logistic loss are bounded by 1. What can you say about the resulting normalized weights for examples that are clearly misclassified in comparison to those that are just slightly misclassified by the current ensemble? If the training data contains mislabeled examples, why do we prefer the logistic loss over the exponential loss, $\text{Loss}(z) = \exp(-z)$?

Solution:

Compute the negative derivative of the logistic loss

$$W_m(t) = -\frac{dL(z)}{dz} = \frac{\exp(-z)}{1 + \exp(-z)} = \frac{1}{\exp(z) + 1} < 1$$

For clearly misclassified examples, $y^t h_m(x^t)$ is a large negative, so $W_m(t)$ is close to 1, while for slightly misclassifed examples, $y^t h_m(x^t)$ is a small negative, so $W_m(t)$ is close to $\frac{1}{2}$. Thus, the normalized weights for the two respective cases will be in a ratio of at most 2:1, i.e., a single clearly misclassifed outlier will never be worth more than two completely uncertain points. This is why boosting with logistic loss is more robust to outliers that with exponential loss.

(b) Suppose the training set is linearly separable and we would use a hard-margin linear support vector machine (no slack) as a base classifier. In the first boosting iteration, what would the resulting $\hat{\alpha}_1$ be?

Solution: In the first step, since the dataset is linearly separable, the hard margin SVM would produce a separator that has $y^t h(x^t; \theta_1) \geqslant 1$ for all t. Then, if we pick α_1 to minimize

$$J(\alpha_1, \theta_1) = \sum_{t=1}^{n} L(\alpha_1 y^t h_1(x^t; \theta_1))$$

it is clear that we need to strictly decrease α_1 because $y^t h_1(x^t; \theta_1) \geqslant 1$. Therefore, the boosting algorithm will set $\alpha = \infty$. This makes sense, because if we can find a base classifier that perfectly separates the data, we will weight it as much as possible to minimize the boosting loss.

(c) Show that we need at most $2n$ stumps to correctly classify n training examples with distinct coordinate values. Consider the case in which the training examples are distinct points in one dimension.

Solution: Consider any n distinct points ($x^i : i = 1, ..., n$) in one dimension. As the points are distinct and n is finite, there exists a positive real $\epsilon > 0$, such that $N_\epsilon(x^i) = (x^i - \epsilon, x^i + \epsilon)$, i.e., the open interval of radius ϵ around each point x^i. Clearly, $N_\epsilon(x^i)$ are all disjoint. Then, for each x^i, consider the two stumps $f_i^+ = sign(y^i(x - (x^i - \epsilon)))$ and $f_i^- = sign(-y^i(x - (x^i + \epsilon)))$. These two stumps perfectly classifies x^i while leaving the other points alone because

$$F_i(x) = \frac{1}{2}f_i^+(x) + \frac{1}{2}f_i^-(x) = y^i, x \in N_\epsilon(x^i) \tag{6.1}$$

$$F_i(x) = \frac{1}{2}f_i^+(x) + \frac{1}{2}f_i^-(x) = 0, x \notin N_\epsilon(x^i) \tag{6.2}$$

It follows that $F(x) = \sum_{i=1}^n F_i(x)$ is an ensemble of $2n$ stumps that allow us to perfectly classify any n distinct points in 1 dimension.

7 Boosting with SVM

For this problem we are given a training set $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of examples and labels. Note that the x_i are vectors. We have no other data available.

We will use boosting as a feature selection method for an SVM classifier. So, we follow the boosting algorithm for m rounds based on D to get m decision stumps $h(x; \hat{\theta}_1), \ldots, h(x; \hat{\theta}_m)$ (we will drop the votes generated by the boosting algorithm). After this we can collect the base classifier predictions into feature vectors

$$\phi(x_t) = \sqrt{m}[h(x_t; \hat{\theta}_1), \ldots, h(x_t; \hat{\theta}_m)]$$

for each training example $t = 1, \ldots, n$. Note that $\|\phi(x_t)\| = 1$.

To train SVM classifiers based on these feature vectors we will split the dataset D into two equal size sets D_{tr} and D_{te}, and use D_{tr} for training and reserve D_{te} for evaluating the performance of the resulting classifier.

(a) Could we use the value of the margin obtained by the hard margin SVM classifiers on D_{tr} as a criterion for selecting between the two kernels below? True or False?

$$K_1(x, x') = \phi(x)^T\phi(x')$$
$$K_2(x, x') = (1 + \phi(x)^T\phi(x'))^2$$

Solution: No. Performance on the training set is not predictive of performance on new data.

(b) Suppose we train a SVM classifier with kernel $K_1(x, x')$ based on D_{tr} and evaluate its performance on D_{te}. Does the performance on D_{te} provide a fair measure of how well the classifier is going to work on yet unseen examples (from the same distribution)? Briefly justify your answer.

Solution: No, classification performance on D_{te} is not a fair measure since the features, the stumps, that the SVM classifier relies on were estimated on the basis of $D = \{D_{te}, D_{tr}\}$, i.e., including D_{te}.

8 Splitting with spheres*

Here is a different way to make decision trees for classification problems in which the inputs $x^{(i)}$ are in $\mathbb{R}^d$. At each node, instead of selecting an attribute x_j to split on, we will select a data point $x^{(i)}$, and will form the split by making a sphere of radius r around that point. So, a test will be of the form

$$\|x - x^{(i)}\| > r$$

where both the example $x^{(i)}$ and the radius r can be chosen. Note that a split describes a sphere in the entire d-dimensional space.

One branch of the decision tree will go on to address the points that are inside the sphere and the other branch will address the points on the outside of the sphere.

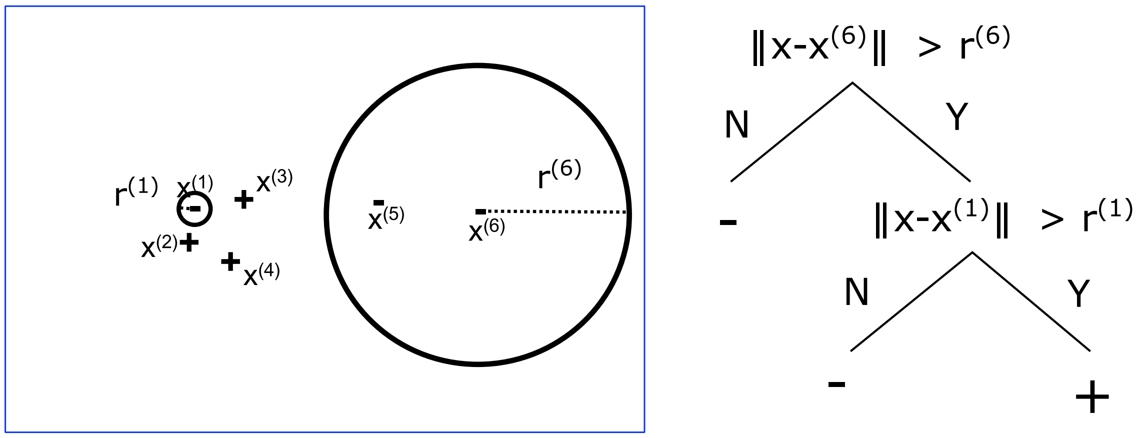
We will assume that the outputs $y^{(i)}$ are in $\{+1, -1\}$.

(a) Is it possible to find a tree with zero training error for any data set in this representation? If so, explain how. If not, explain, why not. Your explanation should fit in the box.

Solution: Yes. You can always end up with a sphere around each individual point with the appropriate label, at the leaves.

(b) Given the data shown below, and using misclassification error to greedily select the splits,

- Draw a decision tree in the blank space to the right of the diagram. Indicate exactly which data points are in each leaf of the tree.
- To show the splits, indicate the $x^{(i)}$ for each test in the internal node of the tree, and indicate the r for each test by drawing the corresponding circle in the data diagram on the left (it's fine if your circles are not perfect).



9 Money tree [8]

You have a decision tree algorithm and you are trying to figure out which attribute is the best to test on first. You are using the information gain metric.

- You are given a set of 128 examples, with 64 positively labeled and 64 negatively labeled.
- There are three attributes, Home Owner, In Debt, and Rich.
- For 64 examples, Home Owner is true. The Home Owner=true examples are 1/4 negative and 3/4 positve.
- For 96 examples, In Debt is true. Of the In Debt=true examples, 1/2 are positive and half are negative.
- For 32 examples, Rich is true. 3/4 of the Rich=true examples are positive and 1/4 are negative.

Use $\log_2$ in your computations.

(a) What is the entropy of the initial set of examples?

Solution: 1.0

(b) What is the information gain of splitting on the Home Owner attribute as the root node?

Solution: Entropy(0.25, 0.75) = 0.811
Gain = 1 - (64/128)*(0.811) - (64/128)*(0.811) = 0.188

(c) What is the information gain of splitting on the In Debt attribute as the root node?

[8]CMU 15-381 Fall 2001 Homework 5, Problem 2

Solution: Entropy(0.5, 0.5) = 1
Gain = 1 - (96/128)*(1) - (32/128)*(1) = 0

(d) What is the information gain of splitting on the Rich attribute as the root node?

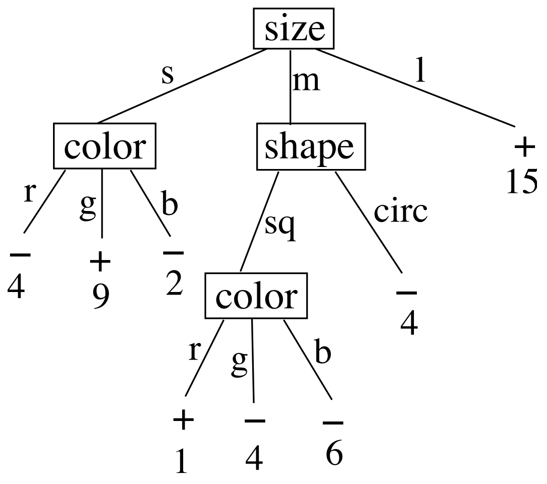
Solution: Entropy(0.25, 0.75) = 0.811
Entropy(0.41, 0.59) = 0.98
Gain = 1 - (32/128)*(0.811) - (96/128)*(0.98) = 0.062

(e) Which attribute do you split on?

Solution: Home Owner, which has the highest information gain.

10 Toy tree [9]

Consider the following decision tree:



Using the above decision tree, classify the following examples. Note that the numbers next to the +/- classification tell how many examples reached that point during the training of the tree.

Size	Color	Shape	Smell	Classification
small	green	square	pine	+
large	blue	square	pine	+
medium	green	circle	rotten egg	-
medium	red	square	lemon	+

[9]CMU 15-381 Fall 2001 Homework 5, Problem 3