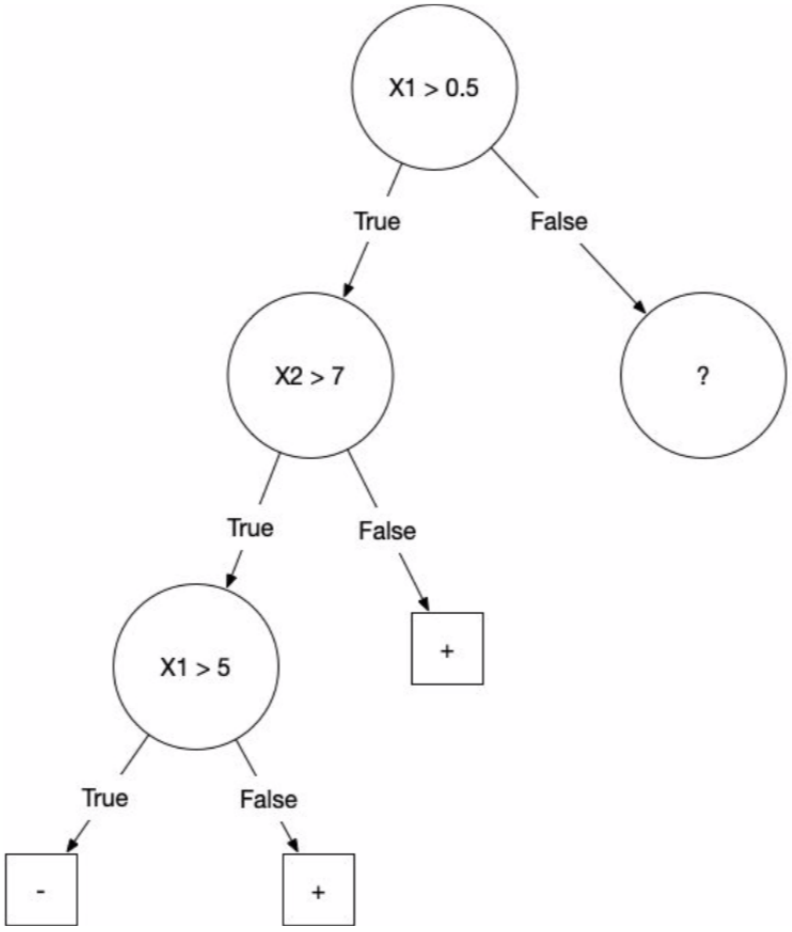
Homework 1 - Supervised Learning Answers

Kasper Gammeltoft

February 20, 2019

1 1. Decision Trees



X	Y
(7.0, 8.0)	-
(3.0, 7.5)	+
(-1.0, 6.5)	+
(-3.0, 6.0)	-
(2.0, 5.5)	+
(-0.5, 3.0)	+
(-2.0, 2.5)	+
(6.5, 1.0)	+
(-0.2, 0.5)	-
(-1.0,-1.0)	-

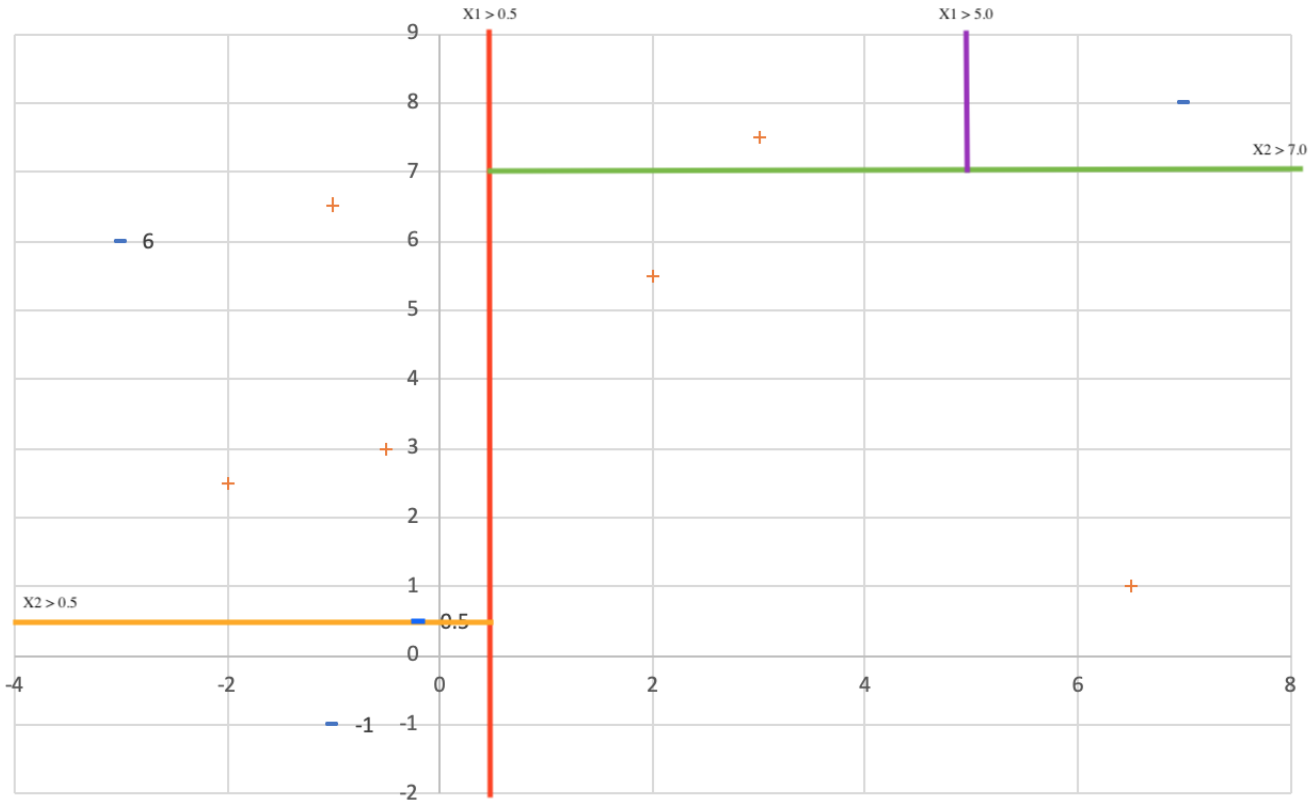
1.1

Info gain with the three possible split choices. Calculated by $H(D) - H(D|A)$ where

$$H(D) = \sum_{y \in D} -p(y_i) \cdot log_2(p(y_i))$$

$X2 > 6.5$	$X2 > 0.5$	$X2 > 0.0$
0	1.57	0.616

1.2



1.3

Misclassification error for three possible split choices.

$X2 > 6.5$	$X2 > 0.5$	$X2 > 0.0$
0.3	0.1	0.2

The split with the largest info gain also has the smallest misclassification error, which is what was expected. As we gain more info per split, we expect less misclassification.

1.4

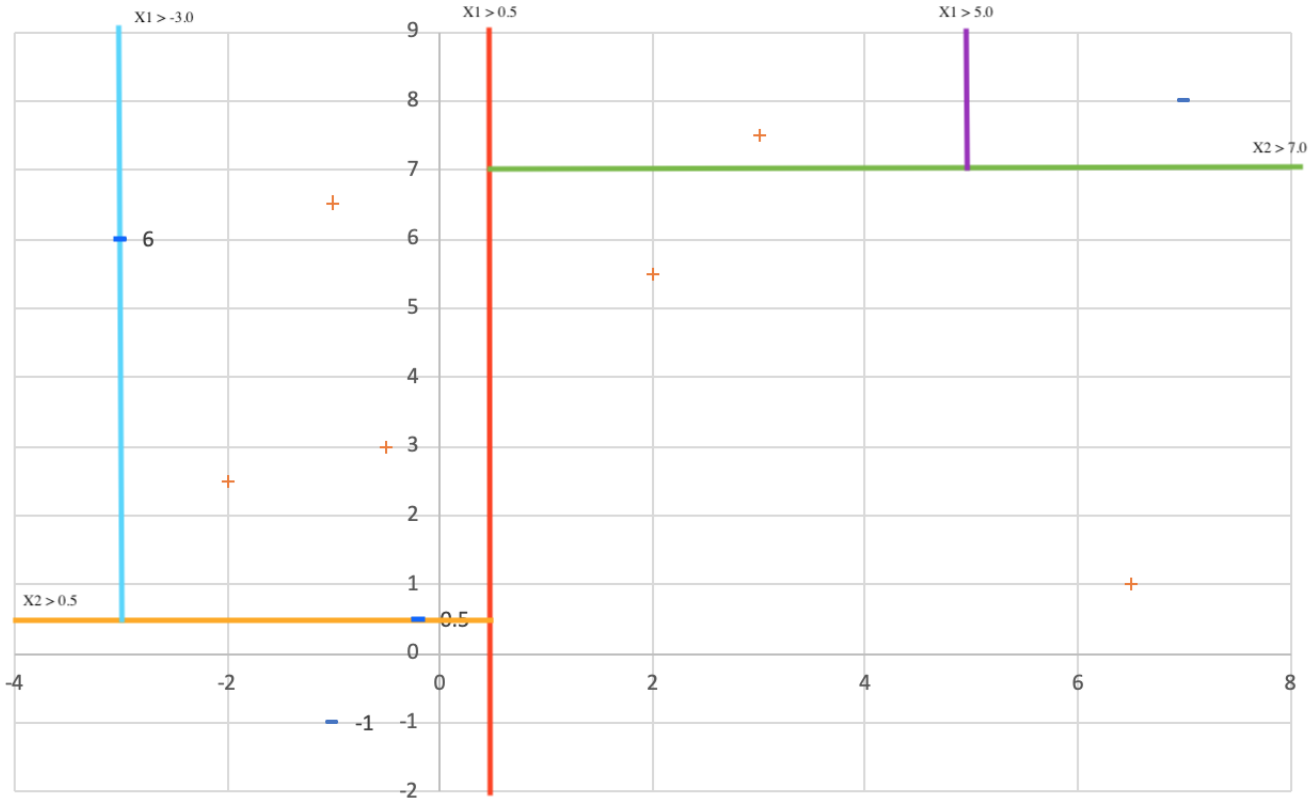
Another option is to consider the minimum and maximum points, then construct cuts for as many data points are along that range. For example, if we only have 3, 5, and 9, construct three possible splits:

1. $\frac{9-3}{3} + 3 = 5$
2. $\frac{9-3}{3} \cdot 2 + 3 = 7$
3. $\frac{9-3}{3} \cdot 3 + 3 = 9$

We then pick the best split from these using the info gained. This does not yield a better result, however. As the data points we have are discrete, picking one of them is always the best policy- as it should be able to classify them just as well as any other point.

1.5

If we pick the split $X1 > -3.0$, then we find we can perfectly classify the data using false as a - and true as a + leaf, after the previous $X2 > 0.5$ is true.



2 2. Polynomial Regression

2.1

There are exactly three ways to achieve features of power 2 with components X_1 and X_2:

X_1^2, X_2^2, and X_1X_2.

2.2

To fit a polynomial of degree 2, we need all of the three features of degree two, plus those of degree one, plus those of degree 0. This would be something of the form:

$$y = b_0 + b_1X_1 + b_2X_2 + b_3X_1^2 + b_4X_2^2 + b_5X_1X_2$$

This sums to be 6 features.

2.3

To generalize, we need to consider first how many features there are of degree n given X is in d dimensions. This comes out to be $\binom{2n-1}{n-1} \cdot \binom{d}{n}$ as we can have only n components from X (if they have *degree* >= 1 for each then any more than n will be too many). We can also see we have $\binom{2n-1}{n-1}$ possibilities for each combination, as for 2 we had $X_1^0X_2^2$, $X_1^1X_2^1$, and $X_1^2X_2^0$, as we count from 0 to 2. This is the multiset number with $n = n$ and $k = n$. Thus, if we need to generalize a polynomial of degree p, we can sum each possible number of features of degree 0 to p given dimensionality d of X. The number of features required is then given by:

$$1 + \sum_{i=1}^{p} \binom{2i-1}{i-1} \cdot \binom{d}{i}$$

3 3 Neural Networks

3.1

We are given the threshold function $g(x)$ and know that our weights are signified by w_j^i, specifying the j^{th} weight of the i^{th} layer. With this weight formation, we can only assume there is one perceptron per layer, or each perceptron will have the same weights for their inputs. Here we will assume each layer has n nodes, and that $|x| = n$. Note that each perceptron has output $y = g(\sum_{j=1}^{n} w_j \cdot x_j)$, where x_j is input from input layer or previous layer.

$h(x)$ for the first layer would thus be $h_{1,j} = g(\sum_{j=1}^{n} x_j w_j^1)$.
$h(x)$ for the second layer would take this as an input: $h_{2,j} = g(\sum_{j=1}^{n} w_j^2 h_{1,j})$.
Thus, plugging in $h_{1,j}$, we get $h(x) = g(\sum_{j=1}^{n} w_j^2 \cdot g(\sum_{j=1}^{n} x_j w_j^1))$.

3.2

Given the error function of means squared: $E(W, x, y) = 1/2(y - h(x))^2$, we can look at the partial derivative given a weight in either the first or second layer. Let's look at the second layer, with $j = 1$: w_1^2.

Plugging in our answer for $h(x)$ above, we get

$$E(W, x, y) = 1/2(y - g(\sum_{j=1}^{n} w_j^2 \cdot g(\sum_{j=1}^{n} x_j w_j^1)))^2$$

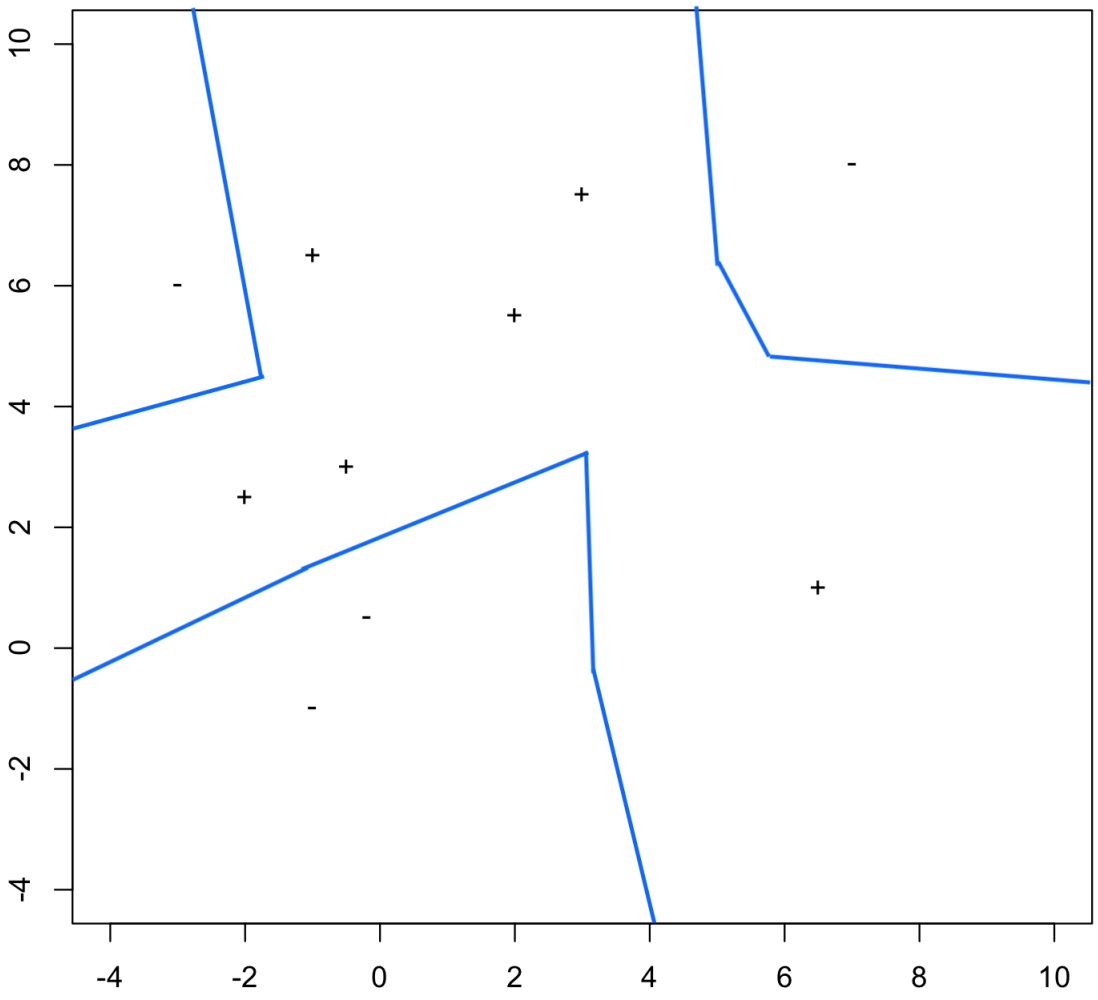
For simplicity, lets take the latter sum as a constant for now: $g(\sum_{j=1}^{n} x_j w_j^1)) = h_1$. Subbing this in and using the chain rule, we get

$$E'(W, x, y) = -2(y - g(\sum_{j=1}^{n} w_j^2 \cdot h_1)) \cdot g'(\sum_{j=1}^{n} w_j^2 \cdot h_1) \cdot h_1$$

Assuming a 'nice' derivitive of g allows this to be done rather easily, something important for neural networks.

4 4 K Nearest Neighbors

4.1



4.2

Misclassification error using leave one out:

K	$Error$
1	0.4
2	0.7
3	0.5
4	0.6
5	0.6

Clearly, $K = 1$ is the best value for K, and has the least error.

5 5 Ensemble Methods

5.1

Generally, bagging does not help with K-Nearest Neighbors. When $K = 1$, all bagging will do is split the smaller $K = 1$ learners amongst th data and average the results, effectively just setting K to be a higher value.

5.2

Just as bagging, in effect, only increases the value of K, boosting will have the opposite effect. As we run the process many times and try to fit points we missed in the last iteration, it will give more weight to those points. This will have the same effect as decreasing K.