Problem Set 1

Suggest a lazy version of the eager decision tree learning algorithm ID3. What are the advantages and disadvantages of your lazy algorithm compared to the original eager algorithm?
Sources:
https://arxiv.org/pdf/1603.02578.pdf

Lazy learning algorithms are similar to KNN in that they do not induce a specific hypothesis from the training set and instead it is delayed until the test set is run. The LazyDT [1] algorithm constructs the best decision tree for each instance as opposed to ID3 which creates a single best decision tree when trained. Batched lazy decision trees behave similarly to lazy decision trees with the difference being that it enables the exploration of an arbitrary sub tree instead of a single path.

Advantages:

- Reduced training times.
- Robust algorithm when it comes to handling missing values in training data.

Disadvantages:

- More storage space required to store the trained models.
- There is no pruning.
- Time to query is long in case of larger number of observations.

Give the VC dimension of these hypothesis spaces, briefly explaining your answers:
1. An origin-centered circle (2D)
2. An origin-centered sphere (3D)

The VC dimension in both cases is 2. For a set of 3 points, r1, r2 and r3, they will be at some point from the origin such that:

r1 <= r2 <= r3

Thus, for 3 points, they cannot be shattered when they are origin centered.

Derive the perceptron training rule and gradient descent training rule for a single unit with output o, where $o = w0 + w1x1 + w1x12 + \ldots + wnxn + wnxn2$. What are the advantages of using gradient descent training rule for training neural networks over the perceptron training rule?

For perceptron training rule, the weights will be modified for each x. Thus,

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = \eta(y - o)x_i$$

Where,
y = target value
o = perceptron output
η = learning rate

For gradient descent with D as the set of training examples, to minimize mean squared error, consider the error function:

$$E(w) \quad = \frac{1}{2} \sum_{d \, \epsilon \, D} (y_d - o_d)^2$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d} (y_d - o_d)^2$$

$$= \frac{1}{2} \sum_{d} \frac{\partial}{\partial w_i} (y_d - o_d)^2$$

$$= \frac{1}{2} \sum_{d} 2(y_d - o_d) \frac{\partial}{\partial w_i} (y_d - o_d)$$

$$= \sum_{d} (y_d - o_d) \frac{\partial}{\partial w_i} (- \sum_{i} \quad (w_i x_i))$$

$$= \sum_{d} (y_d - o_d)(-x_{i,d})$$

$$\Delta w_i = \; -\eta \frac{\partial E}{\partial w_i}$$

Perceptron training rule is guaranteed to succeed if the training examples are linearly separable and a small learning rate. The advantage of gradient descent is that it can deal with non-linearly separable and can deal with noise in the data.

References

1) Friedman, J., Kohavi, R., & Yun, Y. (1996). Lazy decision trees. In Proceedings of the Thirteenth National Conference on Artificial Intelligence (pp. 717–724). Menlo Park, CA: The AAAI Press