Machine Learning CS 726, Fall 2011 *Christoph Schulze, Michael Whidby, and Kotaro Hara*

Project 1: Basic classification

WU1

By computing $mean((datasets.TennisData.Y > 0) == (h.predictAll(datasets.TennisData.X) > 0))$, you are actually computing $\frac{1}{N}\sum_{k=0}^{N-1}[y_k = f(\boldsymbol{x}_k)]$, which, by definition, is accuracy (1 - training error).

WU2

The training accuracy goes down after increasing number of examples. This is because there would be more ambiguity, meaning there would be examples with different lables with same feature values at non-leaf nodes where the DT makes a decision.

The testing error could go down even after you increase the number of examples. This is because the features that are considered important on training phase can vary as the number of examples changes. Let's say, when there are 50 examples, a feature x_{a1} is considered more dominant than the other features. However, as the number of examples increases, say 100, another feature x_{a2} may be considered more influential than x_{a1}. This may be right or wrong on the test data. If it is right on test data, the test accuracy goes up, and vice versa.

WU3

Please refer to the figure 1.

The training accuracy is guaranteed to monotonically increase, because it is table-look-up, except for the case which at every node on DT the probability of going right or left is 50-50. This will never increase the accuracy, but it is very unlikely to happen.

The testing accuracy could oscillate like on the figure, but not guaranteed. As questions asked at DT nodes go up, the model tends to overfit. This could either increase or decrease in accuracy. However, in the end the accuracy goes down overall.

WU4

Please refer to the figure 2.

```
-introduction to low-level programming concepts [212]
   - program analysis and understanding [631]
     - computer processing of pictorial information [733]
       - Leaf -1.0
       - Leaf 1.0
     - introduction to human-computer interaction [434]
       - Leaf -1.0
       - Leaf 1.0
   - computational linguistics ii [773]
```

```
- computational methods [460]
  - Leaf -1.0
  - Leaf 1.0
- computational geometry [754]
  - Leaf 1.0
  - Leaf 1.0
```

Intuitively, the course 212 is not a good feature because it is an introductory course because most students take that course. This means it should not be an important node at the top of the tree. However, there are a few informative questions as well. For example, computational geometry (754) sounds related to graphics, and computational linguistics ii (773) is related to AI. So it makes sense that these questions are asked on the DT, i.e. *correlated* to the event that students take/not take AI or graphics.

However, both 754 and 773 are advanced courses and should be taken after taking AI or graphics course. Since DT cannot take time data (which couses were taken before9 into account in training phase, it cannot explain the *causality*.

WU5

For the course recommender data, generate train/test curves for varying values of K and epsilon (you figure out what are good ranges, this time). Include those curves: do you see evidence of overfitting and underfitting? Next, using K=5, generate learning curves for this data.

Figure 3 shows the training and test plots of KNN with varying k-values. As k approaches N, we are prone to underfitting, since our predictions are more and more based off the average of the examples in the training data. With $k = 1$, we are overfitting possible noise: for example, in the case where a data point we are asked to predict lies in an area of mostly negatives, except for one positive that happens to be the nearest-neighbor.

Figure 4 shows the training and test plots of KNN with various ϵ-values. KNN with ϵ-ball nearest neighbors doesn't fit this problem well, only performing better-than-chance with $\epsilon = 5$. This could suggest that a lot of the data isn't necessarily close in the N-dimensional space.

Figure 5 shows the learning curve for KNN with $k = 5$. In general, we see an improvement in test accuracy as the number of training examples increases. This should be expected, since more training examples will increase the chances of KNN to be able to accurately predict at test time.

WU6

The top and bottom five had weights with the same values so we extended the list from 5 to 7 for the positiv values. We did the same thing for the negative ones where we extended the list from 5 to 6.

Below you can see the top and bottom weights and their corresponding classes.

```
advanced computer graphics
8.0
computer processing of pictorial information
9.0
database management systems
7.0
computational methods
5.0
```

```
computer networks
5.0
data structures
5.0
introduction to information technology
5.0



honors seminar
-8.0
introduction to c programming
-6.0
program analysis and understanding
-5.0
object-oriented programming i
-5.0
object-oriented programming ii
-4.0
computer networks
-4.0
```

There were two things that were interesting in these weights: - no other course from the AI area had a high or low weight associated with it. - there is a clustering of undergrad courses in the negative weights (introduction to c programmin, object-oriented programming i+ii)

Besides that there seemed to be no reason for choosing these weights that we could think of. The positiv weights seem to be randomly drawn from the other categories (besides AI).

Appendix

Figure 1: Plot of DT with various numbers of examples

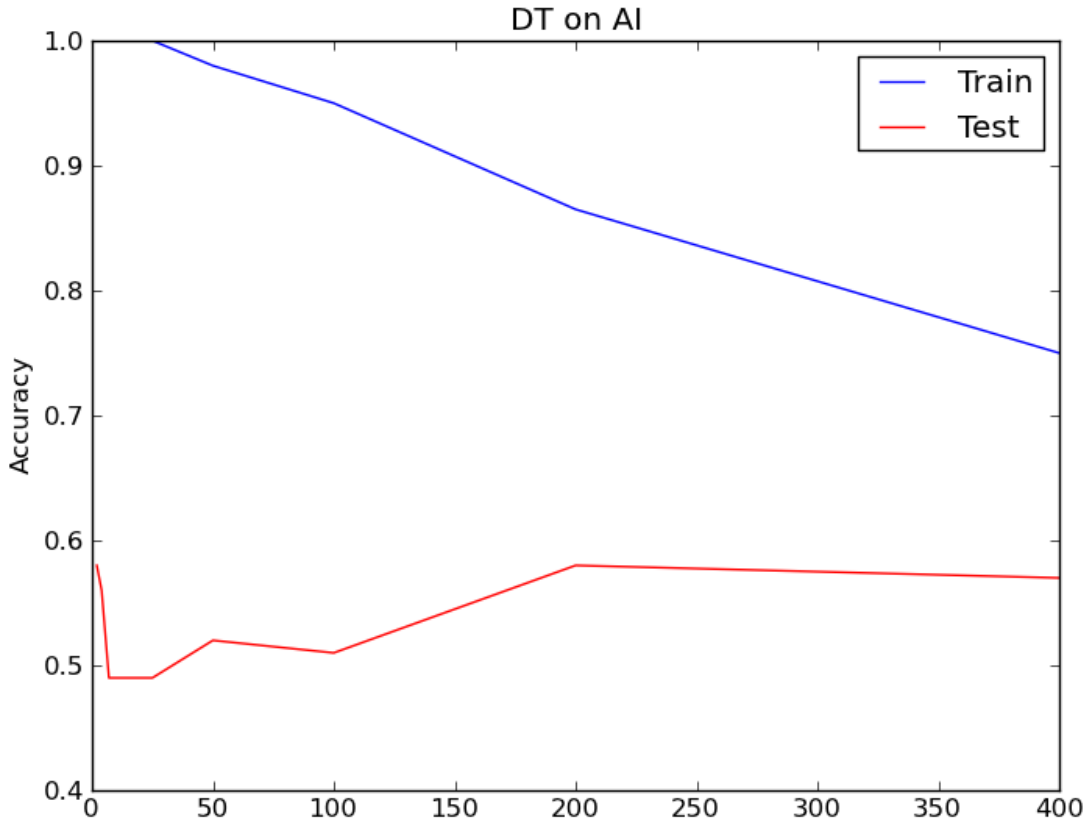

Figure 2: Plot of DT with various depth parameters

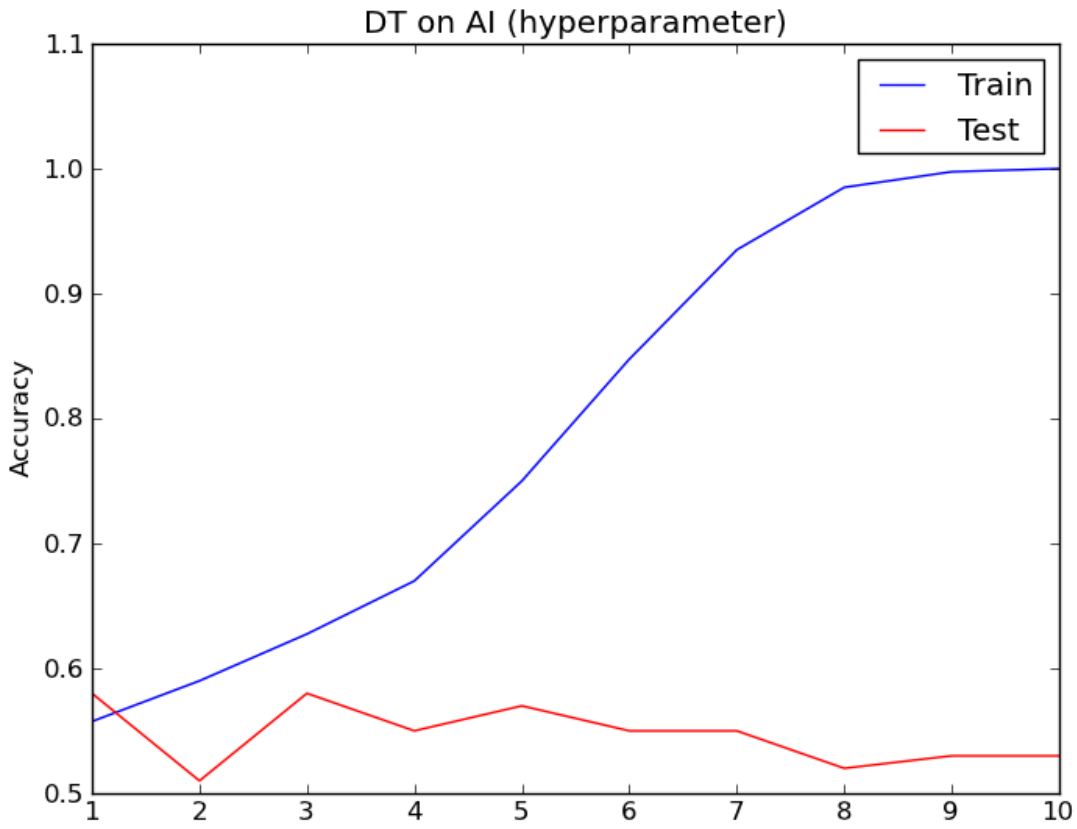

Figure 3: Plot of KNN with various K-values

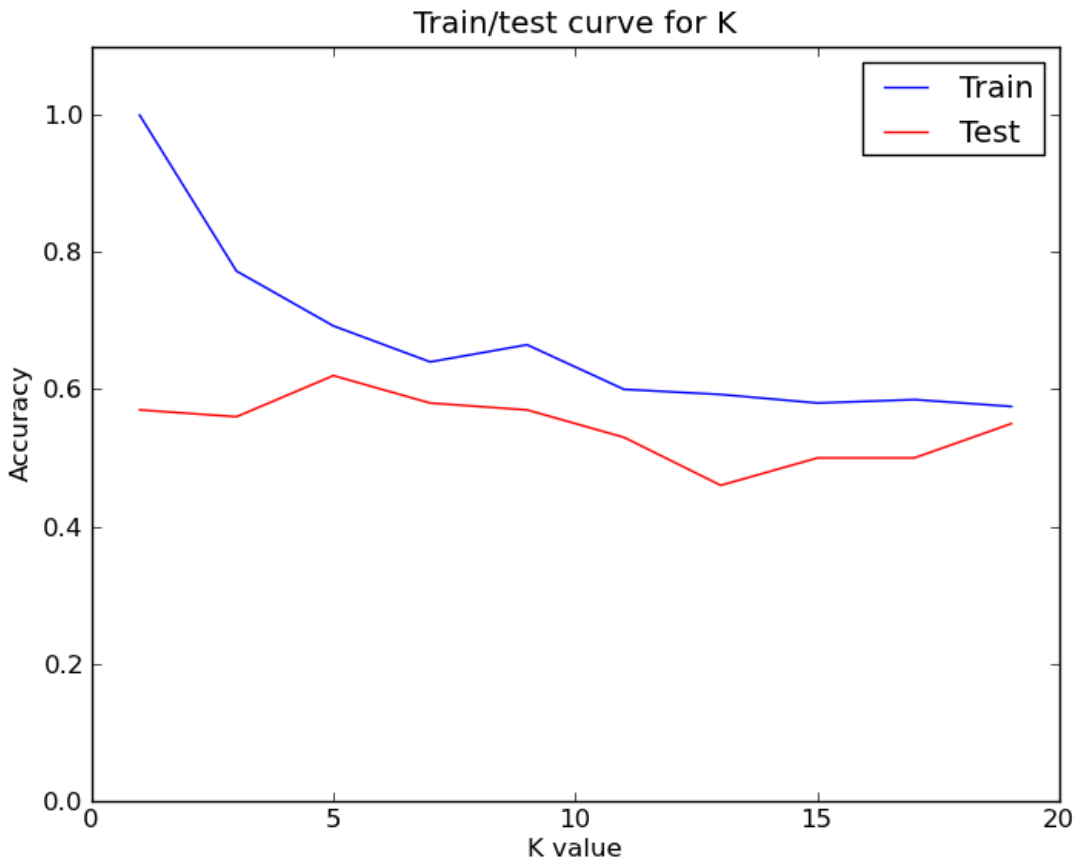

Figure 4: Plot of KNN with various ϵ-values

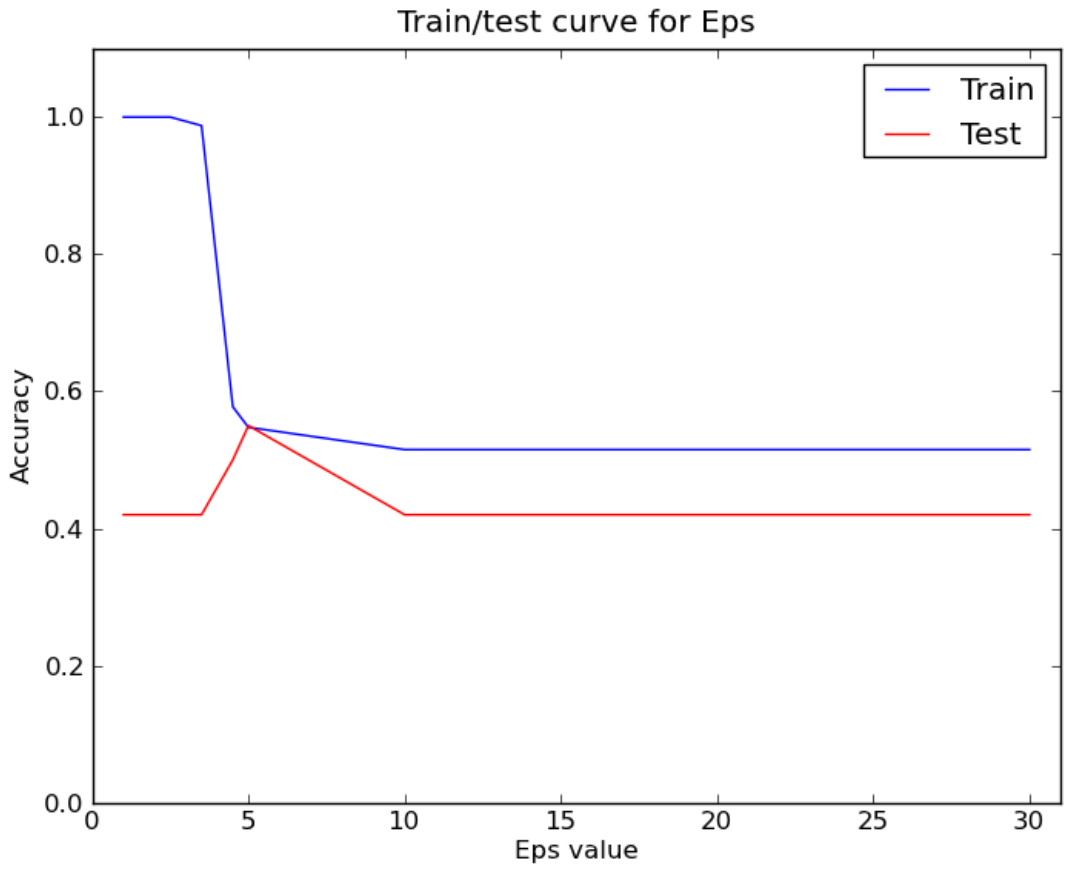

Figure 5: Learning Curve with K=5