Decision Trees

CAC 350

Decision Trees

* Versatile algorithm
* Perform classification and regression
* Fundamental component of Random Forests (might be most powerful ML algorithm)
* White box model - easy to understand, decisions easy to interpret

Example

jupyter

Overfitting

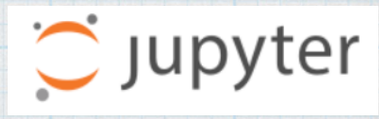
* General property of decision trees
* Easy to go too deep in the tree and fit details of the particular data rather than the overall properties of the distributions
* Let's look at two subsets of the data

Overfitting

* Some places we're consistent while others we are not
* If we use info from both trees, we may have a winner - Random Forests...next algorithm

Iris Example

* Let's use the Iris training set again



Making Predictions

* Start at the root and make your way down to the leaves

* Advantage: Don't require feature scaling or centering

* samples: how many training instances it applies to

Gini impurity

* gini: impurity - a node is "pure" (gini=0) if all training instances it applies to belong to the same class

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

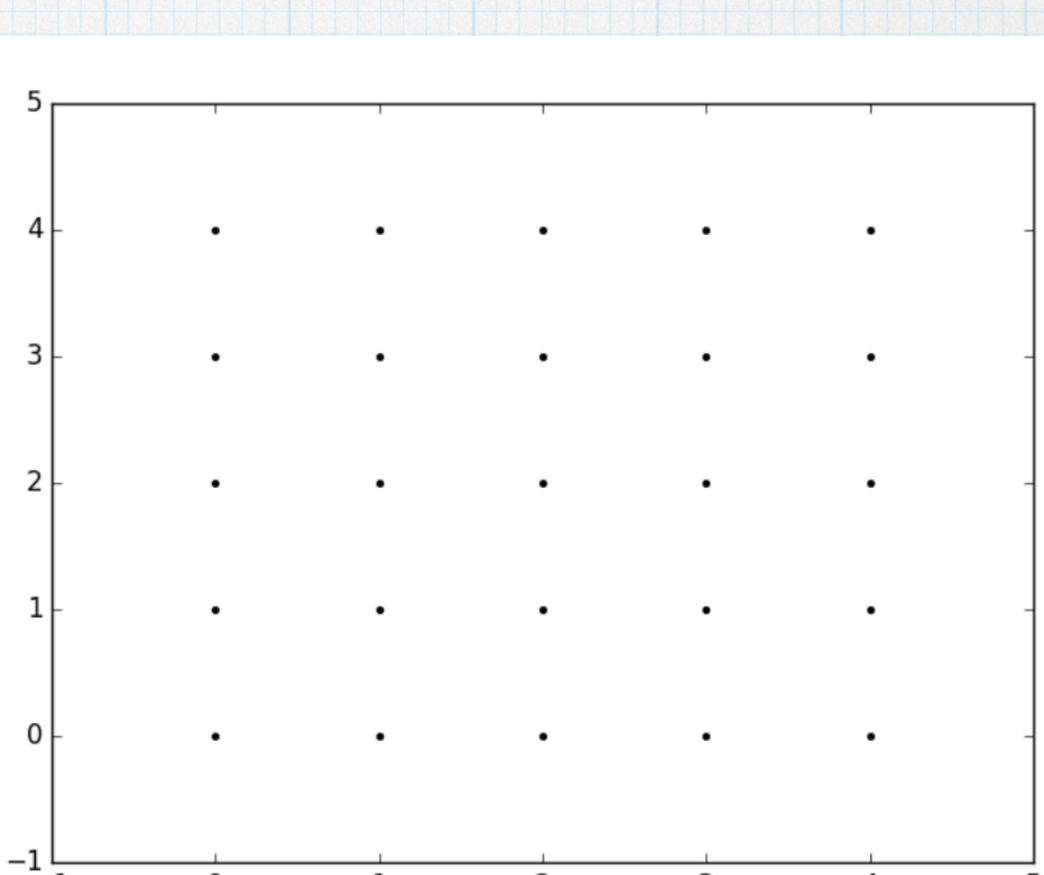
* $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node

np.meshgrid

* Creates a coordinate grid (1-4 example in x and y direction)

```
x[0,0] = 0    y[0,0] = 0
x[0,1] = 1    y[0,1] = 0
x[0,2] = 2    y[0,2] = 0
x[0,3] = 3    y[0,3] = 0
x[0,4] = 4    y[0,4] = 0
x[1,0] = 0    y[1,0] = 1
x[1,1] = 1    y[1,1] = 1
...
x[4,3] = 3    y[4,3] = 4
x[4,4] = 4    y[4,4] = 4
```

```
x =   0 1 2 3 4        y =   0 0 0 0 0
      0 1 2 3 4              1 1 1 1 1
      0 1 2 3 4              2 2 2 2 2
      0 1 2 3 4              3 3 3 3 3
      0 1 2 3 4              4 4 4 4 4
```



np.c_

* Stacks 1D arrays into 2D columns

```
>>> np.c_[np.array([1,2,3]), np.array([4,5,6])]
array([[1, 4],
       [2, 5],
       [3, 6]])
>>> np.c_[np.array([[1,2,3]]), 0, 0, np.array([[4,5,6]])]
array([[1, 2, 3, ..., 4, 5, 6]])
```

jupyter

Estimating Class Probabilities

- DT can estimate the probability that an instance belongs to a particular class k.
- Traverses the tree to find the leaf node for this instance and returns the ration of training instances of class k in this node
- Example: Flower with petals 5cm long, 1.5 cm wide
  - Depth-2 left node > 0% Iris Setosa, 90.7% Iris Versicolor, and 9.3% for Iris Virginica

jupyter

CART Training Algorithm

* Scikit-Learn uses the Classification and Regression Tree (CART) algorithm to train Decision Trees ("growing" trees) - only produces binary trees

  * Splits the training set into two subsets using a single feature k and a threshold t_k ("petal length <= 2.45 cm")

    * How to choose k and t_k? Searches for the pair (k, t_k) that produces the purest subsets (weighted by size)

CART Training Algorithm

* Cost function that the algorithm tries to minimize

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

* where:
  * $G_{left/right}$ measures the impurity of the left/right subset
  * $m_{left/right}$ is the number of instances in the left/right subset

CART Training Algorithm

* Once the CART algorithm has split the training set in two, it splits the subsets using the same logic and then the sub-subsets recursively

* Stops recursion once the max depth is reached - defined by hyper parameter

CART Training Algorithm

* Greedy - searches for optimum split at each level
* Does not take impurity into account
* Reasonably good but not optimal
* Optimal = NP Complete so we settle

Computational Complexity

* Traversing the tree is $O(\log_2(m))$ regardless of number of features
* Hyperparameters: max_features - can limit the number of features the training algorithm compares
  * $O(n * m \log_2(m))$
  * Can speed up if it's a small training set by using presort=True

Gini Impurity or Entropy?

* Another hyperparameter option is to change the criterion to entropy rather than gini

* Remember: gini impurity is the measure or probability of a particular variable being improperly classified when randomly chosen

* Entropy (when applied to information theory - Claude Shannon): entropy = 0 when all messages are identical

* ML: a set's entropy = 0 when it contains instances of only one class

Which is better?

Gini Impurity

* Slightly faster
* Isolates most frequent class in its own branch

Entropy

* More balanced trees

Regularization Hyperparameters

* Nonparametric model - number of parameters is not determined prior to training allowing the model to stick closely to the data, often resulting in overfitting

* Parametric model (linear model) - has predetermined number of parameters limiting degree of freedom

    * Less risk of overfitting, higher risk of underfitting

Random Forest

* RandomForestClasifier: Convenient and optimized for Decision Trees
* RandomForestRegressor: used for regression tasks as decision trees can also be used for regression purposes
* Hyperparameters control how the trees are grown
* RF algorithm introduces extra randomness by searching for the best feature among a random subset of features rather than searching for the very best feature when splitting a node

What is a hyperparameter?

* Parameter of the learning algorithms
* Must be set prior to training and remains constant during training

How do you know which hyperparameter to choose?

* Train, validation, test split
* Train on 100 different models using 100 different hyperparameter values, choose the best one
* Cross-validation: many small validation sets
  * Each model is evaluated once per validation set after it is trained on the rest of the data
  * More accurate measure, but increased training time

Hyperparameter Importance

* Random forests can be used to assess attribute importance: if removing an attribute from the dataset yields a drop in performance - what does that mean?

* Forward selection: predicts the performance of a classifier based on a subset of hyper parameters that is initialized empty and greedily filled with the next most important hyper parameter

Hyperparameter Importance

* Ablation Analysis: requires a default setting and an optimized setting and calculates a so-called ablation trace - embodies how much the hyperparameters contributed towards the difference in performance between the two settings

* Functional ANOVA: can detect the importance of both individual hyperparameters and interaction effects between arbitrary subsets of hyperparameters across datasets

Article on Moodle

* Shows how functional ANOVA...
  * is computationally more efficient than forward selection
  * can detect interaction effects
  * does not rely on a specific default configuration

Techniques for Optimizing Hyperparameters

* Random search
* Bayesian Optimization
* Evolutionary Optimization
* Meta-Learning
* Bandit-Based Methods

The ML Recipe

1. Choose a class of model - what are you trying to do??? Linear model (prediction) or classification?

2. Choose model hyperparameters - a class of a model is not the same as an instance of a model

Hyperparameters

* Once you have the model class, think about...
    * Would we like to fit for the offset (i.e., intercept) (simple corrections to the response column)?
    * Would we like the model to be normalized (changing data to a common scale without distorting range of values)?
    * Would we like to preprocess our features to add model flexibility?
    * What degree of regularization would we like to use in our model (discourages learning a more complex or flexible model, so as to avoid the risk of overfitting)?
    * How many model components would we like to use?

The ML Recipe

3. Arrange data into a features matrix and target vector

4. Fit the model to your data

5. Predict labels for unknown data

6. Validate the model - need to demonstrate that the model and the hyper parameters are a good fit to the data

Optional Exercise

* Work with 1-2 partners
* Numbers 7 and 8 in Chapter 6 of Hands-on Machine Learning