Data Representation
Testing and evaluation schemes
Labs and Tools

Data Set Features

- Data Types
  - Nominal (aka Categorical, Discrete)
  - Continuous (aka Real, Numeric)
  - Linear (aka Ordinal) – Is usually just treated as continuous, so that ordering info is maintained
- Consider a Task: Classifying the quality of pizza
  - What features might we use? Do one of each versions above.
- How to represent those features?
  - Will usually depend on the learning model we are using
- *Classification* assumes the output class is nominal. If output is continuous, then we are doing *regression*.

Fitting Data to the Model

- Continuous -> Nominal
  – Discretize into bins – more on this later
- Nominal -> Continuous (Perceptron expects continuous)
  a) One input node for each nominal value where one of the nodes is set to 1 and the other nodes are set to 0 – One Hot
     - Can also *explode* the variable into n-1 input nodes where the most common value is not explicitly represented (i.e. the all 0 case)

  b) Use 1 node but with a different continuous value representing each nominal value
  c) Distributed – $\log_b n$ nodes can uniquely represent n nominal values (e.g. 3 binary nodes could represent 8 values)
  d) If there is a very large number of nominal values, could cluster (discretize) them into a more manageable number of values and then use one of the techniques above
- Linear data is already in continuous form

Data Normalization

- What would happen if you used two input features in an astronomical task as follows:
  - Weight of the planet in grams
  - Diameter of the planet in light-years

Data Normalization

- What would happen if you used two input features in an astronomical task as follows:
  – Weight of the planet in grams
  – Diameter of the planet in light-years
- Normalize the Data between 0 and 1 (or similar bounds)
  – For a specific instance, could get the normalized feature as follows:

$$f_{normalized} = (f_{original} - Minvalue_{TS})/(Maxvalue_{TS} - Minvalue_{TS})$$

- Use these same Max and Min values to normalize data in novel instances
- Sklearn has methods to do this and other normalization approaches
- Note that a novel instance may have a normalized value outside 0 and 1
  – Why? Is it a big issue?

ARFF Files

- An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a Machine Learning dataset (or relation).
  - Developed at the University of Waikato (NZ) for use with the Weka machine learning software (http://www.cs.waikato.ac.nz/~ml/weka).
  - We will commonly use the ARFF format for CS 270
- ARFF files have two distinct sections:
  - Metadata information
    - Name of relation (Data Set)
    - List of attributes and domains
  - Data information
    - Actual instances or rows of the relation
- Optional comments may also be included which give information about the Data Set (lines prefixed with %)

Sample ARFF File

```
% 1. Title: Pizza Database
% 2. Sources:
%      (a) Creator: BYU CS 270 Class…
%      (b) Statistics about the features, etc.

@RELATION Pizza

@ATTRIBUTE Weight        real
@ATTRIBUTE Crust         {Pan, Thin, Stuffed}
@ATTRIBUTE Cheesiness    real
@ATTRIBUTE Meat          {True, False}
@ATTRIBUTE Quality       {Good, Great}

@DATA
.9,    Stuffed,    99,    True,     Great
.1,    Thin,       2,     False,    Good
?,     Thin,       60,    True,     Good
.6,    Pan,        60,    True,     Great
```

- Any column could be the output, but we will assume that the last column(s) is the output
- Assume cheesiness is linear (an integer between 0 and 100)
- What would you do to this data before using it with a perceptron and what would the perceptron look like? – Show an updated ARFF row

ARFF Files

- More details and syntax information for ARFF files can be found at our website
- Also have a small arff library to help you out
- Data sets that we have already put into the ARFF format can also be found at our website and linked to from the LS content page

http://axon.cs.byu.edu/data/

- You will use a number of these in your simulations throughout the semester – Always read about the task, features, etc, rather than just plugging in the numbers
- You will create your own ARFF files in some projects, and particularly with the group project

Performance Measures

- There are a number of ways to measure the performance of a learning algorithm:
  - Predictive accuracy of the induced model (or error)
  - Size of the induced model
  - Time to compute the induced model
  - etc.
- We will focus mostly on accuracy/error
- Fundamental Assumption:

Future novel instances are drawn from the same/similar distribution as the training instances

Training/Testing Alternatives

- Four methods that we commonly use:
  - Training set method
  - Static split test set
  - Random split test set CV
  - *N*-fold cross-validation
  - The last two are the more accurate approaches

Training Set Method

- Procedure
  - Train model with the training set
  - Compute accuracy on the same training set
- Simple but least reliable estimate of future performance on unseen data (a rote learner could score 100%!)
- Not used as a performance metric but it is often important information in understanding how a machine learning model learns
- This is information which you will often report in your labs and then compare it with how the learner does with a better method

Static Training/Test Set

- Static Split Approach
  - The data owner makes available to the machine learner two distinct datasets:
    - One is used for learning/training (i.e., inducing a model), and
    - One is used exclusively for testing
- Note that this gives a way to do repeatable tests
- Can be used for challenges (e.g. to see how everyone does on one particular unseen set, method we use for helping grade your labs.)
- Be careful not to overfit the Test Set ("Gold Standard")

Cross-Validation (CV)

- Cross-Validation (CV) – Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations
- We then average the results of these iterations
- With CV we avoid having data only used for either training or test, and give all data a chance to be part of each, thus getting more accurate results

Random Training/Test Set Approach

- Random Split CV Approach (aka holdout method)
  - The data owner makes available to the machine learner a single dataset
  - The machine learner splits the dataset into a training and a test set, such that:
    - Instances are randomly assigned to either set
    - The distribution of instances (with respect to the target class) is hopefully similar in both sets due to randomizing the data before the split
      - Stratification is an option to ensure proper distribution
    - Typically 60% to 90% of instances is used for training and the remainder for testing – the more data there is the more that can be used for training and still get statistically significant test predictions
    - Once is rarely enough. Could get a lucky or unlucky test set
  - Do multiple training runs with different random splits and average the results to get a more statistically accurate prediction of generalization accuracy.

N-fold Cross-validation

- Use all the data for both training and testing
  - More structured than the random train/test split approach
  - Each instance is tested exactly once and used for training N-1 times
- Procedure
  - Partition the randomized dataset (call it D) into N equally-sized subsets $S_1, \ldots, S_N$
  - For $k = 1$ to N
    - Let M_k be the model induced from $D - S_k$
    - Let a_k be the accuracy of M_k on the instances of the test fold S_k
  - Return $(a_1+a_2+\ldots+a_N)/N$

N-fold Cross-validation

Iteration 1	Test	Train	Train	Train	Train
Iteration 2	Train	Test	Train	Train	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Train	Train	Test	Train
Iteration 5	Train	Train	Train	Train	Test

Record test accuracy at each iteration and report the average

N-fold Cross-validation (cont.)

- The larger *N* is, the smaller the variance in the final result
- Commonly, a value of *N*=10 is considered a reasonable compromise between time complexity and reliability
- Still must chose an actual model to use during execution – how?

N-fold Cross-validation (cont.)

- The larger *N* is, the smaller the variance in the final result
- Commonly, a value of *N*=10 is considered a reasonable compromise between time complexity and reliability
- Still must chose an actual model to use during execution - how?
  - Should we select the one model that was best on its test fold?
  - All data! With any of the approaches we have presented
- Note that N-fold CV is just a better way to estimate how well we will do on novel data, rather than a way to do *model selection*

Machine Learning Tools

- Lots of new Machine Learning Tools
  - Weka was the first main site with lots of ready to run models
  - Scikit-learn now very popular
  - Languages:
    - Python with NumPy, matplotlib, Pandas, other libraries
    - R (good statistical packages), but with growing Python libraries…
  - Deep Learning Neural Network frameworks – GPU capabilities
    - Tensorflow - Google
    - PyTorch – Multiple developers (Facebook, twitter, Nvidia...) - Python
    - Others: Caffe2 (Facebook), Keras, Theano, CNTK (Microsoft)
  - Data Mining Business packages – Visualization, Expensive
- Great for experimenting and applying to real problems
- But important to "get under the hood" and not just be black box ML users

Doing Your Labs

- We will use scikit-learn for individual labs
  - Whatever you want in group project
- Program in Python in Jupyter notebooks
  - NumPy library – Great with arrays, etc.
- Recommended tools and libraries
  - Colab – Google IDE for Python and Jupyter notebooks
  - Pandas – Data Frames and tools are very convenient
  - MatplotLib

scikit-learn (SK)

- One of the most used and powerful machine learning toolkits out there
- Lots of implemented models and tools to use for machine learning applications
  - Sometimes missing some things we would like, but it is continually evolving
  - Source is available, and you can always override methods with your own, etc.
- Basically a Python Library to call from your Python code
- Familiarize yourself with the scikit-learn website as you will be using it for all labs

Perceptron Project

- Content Section of LS (Learning Suite) for project specifications
  - Review carefully the introductory part regarding all projects
- For each project carefully read the specifications for the lab in the Jupyter notebook on GitHub
- You can just copy the Perceptron notebook from the GitHub site to your computer and then add your work in the code and text boxes