Feature Engineering Part 2: Comprehensive Review

DSC 208R - Data Management for Analytics

Dimensionality Reduction

Dimensionality reduction techniques are often used on relational/structured/tabular data to transform features into a different, typically lower-dimensional, latent space.

Key Concepts

- **Basic Idea**: Transforms features into a new set of features, often linear combinations of the originals, to reduce the number of features while retaining important information.
- **Examples**: Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Latent Dirichlet Allocation (LDA), Matrix factorization.
- **Distinction from Feature Selection**:
  - *Feature Selection*: Preserves the original semantics of each feature, simply selecting a subset of existing features.
  - *Dimensionality Reduction*: Typically does *not* preserve the semantics of individual features, as new features are combinations of old ones in "nonsensical" (but mathematically meaningful) ways.
- **Scalability**: Scaling dimensionality reduction techniques can be non-trivial, similar to scaling individual machine learning training algorithms.

Time-Based Features

Many relational or tabular datasets include time/date attributes which can be rich sources of features.

- **Extraction from Datetime Columns**: Per-example reconversion to extract numerical (e.g., year, month, day of week, hour) or categorical features (e.g., season, holiday).
- **Global Statistics for Calibration**: Sometimes, global statistics are needed to calibrate time-based features (e.g., normalizing by the maximum value of a time series).
- **Complex Temporal Features**: More complex temporal features are studied extensively in time series mining, such as lags, moving averages, or trends.

Text-Based Features (Review)

Text data requires specific feature engineering techniques to convert unstructured text into a numerical format suitable for ML models.

- **Bag of Words (BoW)**: Represents text as an unordered collection of word counts. Simple but loses word order and context.
- **TF-IDF (Term Frequency-Inverse Document Frequency)**: Weights words based on their frequency in a document and their rarity across the corpus, providing a more informative representation than raw counts.
- **Word Embeddings (e.g., Word2Vec, GloVe)**: Dense vector representations that capture semantic relationships between words based on their co-occurrence patterns in large text corpora.

- **N-grams**: Sequences of N words (or characters) used to capture some local word order and context.
- **Implementation in MapReduce/Spark**:
  - *Bag of Words/N-grams*: Can be implemented as a Map-only function, where each document is mapped to its term/n-gram counts.
  - *TF-IDF*: Requires both Map and Reduce stages. Map calculates term frequency per document. Reduce (or a second MapReduce pass) calculates inverse document frequency using global document counts.

Learned Feature Extraction in Deep Learning

Deep Learning (DL) models offer a significant advantage by automating the feature engineering process, especially for unstructured data.

- **No Manual Feature Engineering**: A major benefit of DL is that it largely eliminates the need for manual feature engineering on unstructured data (e.g., images, text, audio).
- **Not Common on Tabular Data**: DL is generally not common or highly effective for structured/tabular/relational data compared to traditional ML models, where manual feature engineering remains crucial.
- **Versatile Data Types**: DL is very versatile, handling almost any data type as input and/or output:
  - *Convolutional Neural Networks (CNNs)*: Used over image tensors.
  - *Transformers and Recurrent Neural Networks (RNNs)*: Used over text.
  - *Graph Neural Networks (GNNs)*: Used over graph-structured data.
- **Internal Feature Transformation**: The neural network architecture itself specifies how to extract and transform features internally through layers, with weights that are learned during the training process.
- **Software 2.0**: This concept refers to programs where features and logic are "learned" directly from data (like in DL) rather than being hand-crafted by human programmers.

Hyper-Parameter Tuning (HT)

Hyper-parameter tuning is an essential "outer loop" around the machine learning model training and inference process.

Definition and Examples

- **Definition**: Hyper-parameters are "knobs" for an ML model or its training algorithm that control the bias-variance tradeoff in a dataset-specific manner to make learning effective.
- **Examples**:
  - *Generalized Linear Models (GLMs)*: L1 or L2 regularization strength.
  - *All Gradient Methods*: Learning rate.
  - *Mini-batch Stochastic Gradient Descent (SGD)*: Batch size.

Common Approaches

- **Grid Search**: The most common approach, where a set of discrete values is chosen for each hyper-parameter, and the model is trained and evaluated on all possible combinations (Cartesian product).
- **Random Search**: Another common approach where values for hyper-parameters are sampled randomly from defined distributions or ranges. Often more efficient than grid search in high-dimensional hyper-parameter spaces.
- **Automated ML (AutoML)**: Complex heuristics and algorithms exist for automated hyper-parameter tuning and even neural architecture search, reducing manual effort but often computationally intensive.

HT involves training and testing multiple model configurations and consuming the results to find the optimal set of hyper-parameters.