Scribing: Feature Selection & Dimension Reduction

LAU, Trelan
SID:1155214344

Contents

1 Introduction to Feature Selection and Dimension Reduction

In modern data analysis, especially with biological data, datasets can be massive. Gene expression profiles, for instance, can involve tens of thousands of genes for thousands of cells, resulting in enormous matrices (slides 21-22). This high dimensionality brings several challenges:

- **Data Size:** Large datasets are computationally expensive to store and process. A 63.5 million cell dataset with 25,000 genes could reach over 5 Terabytes (slide 22).
- **Noise and Redundancy:** Datasets often contain irrelevant features (noise) or highly correlated features (redundancy), which can degrade the performance of machine learning models.
- **Curse of Dimensionality:** As the number of features grows, the data becomes increasingly sparse, making it difficult to find meaningful patterns.

Feature selection and dimension reduction are techniques used to address these challenges by creating a smaller, more manageable set of features while retaining as much valuable information as possible (slides 24, 25).

1.1 Key Benefits

The primary goals and benefits of applying these techniques include (slide 26):

- **Data Compression:** Reduces storage requirements and computational cost.
- **Improved Prediction Performance:** Removes irrelevant and redundant data that can mislead learning algorithms.
- **Enhanced Understanding:** Simplifies models and allows for better interpretation of the underlying factors driving predictions (e.g., identifying genes related to a specific cancer).
- **Better Data Visualization:** Facilitates the visualization of high-dimensional data in 2D or 3D, making it easier to identify patterns and clusters.

2 Model Evaluation Techniques

Before reducing dimensionality, it is crucial to understand how to evaluate the performance of a model. The choice of evaluation metric is critical and depends on the specific task, such as classification or clustering.

2.1 Binary Classification Evaluation

In binary classification, where there are two possible outcomes (e.g., "Yes" or "No"), we often use a confusion matrix to evaluate performance (slide 4).

2.1.1 Key Metrics

- **Accuracy:** The proportion of correctly classified instances. While simple, it can be misleading for imbalanced datasets.
- **Precision:** The proportion of true positive predictions among all positive predictions. It answers: "Of all instances predicted as positive, how many were actually positive?"

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall (Sensitivity):** The proportion of actual positives that were correctly identified. It answers: "Of all the actual positive instances, how many did we correctly predict?"

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **F1-score:** The harmonic mean of precision and recall, providing a single score that balances both metrics.
$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
- **Balanced Accuracy:** Averages the recall for each class, which is useful for imbalanced datasets.

Context is crucial. For a rare cancer screening test, a high recall is more important than precision, as failing to identify a sick person (a false negative) is far more dangerous than incorrectly flagging a healthy person (a false positive) (slide 4).

2.2 Multi-class Classification and Evaluation

For tasks with more than two classes, the problem can be treated as a series of binary classification problems (one-vs-rest). Metrics like accuracy, precision, and recall are still used but must be aggregated into a single score (slide 9).

- **Macro-average:** Calculates the metric independently for each class and then takes the average. It treats all classes equally, regardless of their size. The performance of small classes has a significant impact on the final score.
$$\text{Macro-average} = \frac{\sum_{i=1}^{C} \text{Metric}_i}{C}$$
- **Micro-average:** Aggregates the contributions of all classes to compute the average metric. It is equivalent to the overall accuracy and is weighted by the number of instances in each class.
$$\text{Micro-average} = \frac{\sum_{i=1}^{C} \text{TP}_i}{\sum_{i=1}^{C} (\text{TP}_i + \text{FP}_i)}$$

2.3 Cross-Validation

Cross-validation is a robust technique for assessing how a model will generalize to an independent dataset.

- **5-fold Cross-Validation (slide 5):** The data is split into 5 equal folds. The model is trained on 4 folds and tested on the remaining fold. This process is repeated 5 times, with each fold used as the test set once. The final performance is the average of the results from the 5 iterations.
- **Leave-one-out Cross-Validation (slide 6):** A special case of n-fold cross-validation where n equals the number of data points. For each iteration, one data point is used for validation, and the rest are used for training. This is computationally expensive but provides an almost unbiased estimate of the test error.

2.4 Clustering Evaluation

Clustering is different from classification because there are no predefined labels. The goal is to evaluate whether similar items are grouped together. This is achieved by evaluating pairs of cells (slide 12). A confusion matrix can be adapted to this task, considering pairs of data points.

2.4.1 Rand Index

The Rand Index (RI) measures the similarity between two data clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. The four quantities are (slide 13):

- **a (TP):** Number of pairs in the same cluster in both true and predicted clusterings.
- **b (FN):** Number of pairs in the same cluster in the true clustering but in different clusters in the predicted clustering.
- **c (FP):** Number of pairs in different clusters in the true clustering but in the same cluster in the predicted clustering.
- **d (TN):** Number of pairs in different clusters in both true and predicted clusterings.

The Rand Index is calculated as follows (slide 14):

$$\text{RI} = \frac{a+d}{a+b+c+d} = \frac{\text{TP} + \text{TN}}{\text{Total number of pairs}}$$

The total number of pairs is given by $\binom{n}{2} = \frac{n(n-1)}{2}$, where n is the total number of data points.

3 Feature Selection Techniques

Feature selection involves choosing a subset of the original features without any transformation. It aims to select the most relevant features to build simpler and more effective models. It can be a supervised or unsupervised process (slide 29).

3.1 Feature Ranking

Feature ranking is the process of evaluating and ordering individual features based on certain criteria, such as their correlation with the class label or mutual information (slide 32). However, ranking features individually has limitations (slides 33, 34):

- **Redundancy:** Ranking may select multiple highly correlated features (e.g., height and weight), which provide similar information.
- **Feature Interaction:** A feature that is not useful by itself may become highly valuable when combined with another feature (e.g., occupation and age to predict salary).

Therefore, selecting the top k individual features does not guarantee the best feature subset of size k.

3.2 Subset Feature Selection

Subset selection methods aim to find the optimal combination of features.

- **Filter Methods (slide 35):** These methods select features based on their intrinsic properties, independent of the learning algorithm. They are computationally fast.
  - **Variance Threshold:** Features with low variance carry less information and can be removed.
  - **Information Gain:** Selects features that provide the most information about the target variable.
- **Wrapper Methods (slide 35):** These methods use a predictive model to score feature subsets. The selection process is "wrapped" around the model, making it computationally expensive but often leading to better performance.
  - **Sequential Feature Selection:** Features are sequentially added to an empty set (forward selection) or removed from a full set (backward elimination) until the optimal subset is found. The performance is typically evaluated using cross-validation (slides 37, 38).

4 Dimension Reduction Techniques

Dimension reduction transforms the original high-dimensional data into a lower-dimensional space. The new features are combinations of the original ones and may not have a direct physical meaning (slides 30, 41).

4.1 Principal Component Analysis (PCA)

PCA is an unsupervised linear transformation technique that creates a new coordinate system for the data. The new axes, called principal components, are orthogonal and are ordered by the amount of variance they explain in the data (slides 43-46).

- The first principal component (PC1) is the direction that captures the maximum variance in the data.

- Each subsequent component captures the maximum remaining variance while being orthogonal to the previous components.

By projecting the data onto the first few principal components, we can reduce dimensionality while preserving most of the original information (variance).

4.1.1 Steps to Perform PCA (slides 47-60)

1. **Normalize the Data:** Standardize each feature so that it has a mean of 0. This is crucial if features are on different scales.
2. **Calculate the Covariance Matrix:** Compute the covariance matrix (Σ) of the normalized data matrix (X').
$$\Sigma = \frac{1}{n-1} X'^T X'$$
3. **Find Eigenvectors and Eigenvalues:** Compute the eigenvectors and eigenvalues of the covariance matrix. The eigenvectors represent the principal components (directions of maximum variance), and the eigenvalues represent the amount of variance captured by each principal component.
4. **Select Principal Components:** Sort the eigenvalues in descending order and choose the top M eigenvectors corresponding to the M largest eigenvalues. These form the new feature space.
5. **Project the Data:** Transform the original normalized data onto the new subspace defined by the selected eigenvectors to obtain the lower-dimensional data ($\hat{X}$).
$$\hat{X} = X'P$$
where P is the matrix of the top M eigenvectors.

5 Implementation in Python

Python's `scikit-learn` library provides powerful tools for both feature selection and dimension reduction.

- **Feature Selection:** The `sklearn.feature_selection` module offers classes like `VarianceThreshold` for filter methods (slide 63).
- **Dimension Reduction (PCA):** The `sklearn.decomposition.PCA` class allows for easy implementation of PCA. The `explained_variance_ratio_` attribute is useful for seeing how much variance is captured by each principal component (slide 63).
- **Model Evaluation:** The `sklearn.metrics` and `sklearn.model_selection` modules provide functions for `classification_report`, `rand_score`, and `cross_val_score` (slides 18, 19).

Listing 1: PCA in Python using scikit-learn

```
import numpy as np
from sklearn.decomposition import PCA

# Sample Data
X = np.array([[-1, -1], [-2, -1], [-3, -2],
              [1, 1], [2, 1], [3, 2]])

# Initialize and fit PCA
pca = PCA(n_components=2)
pca.fit(X)

# Check explained variance
print(pca.explained_variance_ratio_)
# Output: [0.9924... 0.0075...]
```