About the PCA

1 The details of PCA

In all the machine learning tasks in this article, we will apply Principal Component Analysis (PCA) [1] on the feature matrix $\mathbf{X}$ and retain a proportion z of the variance, we follow these four steps. The first step is to standardize $\mathbf{X}$ to obtain $\mathbf{X}_{\text{std}}$, ensuring each feature has a mean of 0 and a variance of 1. This standardization removes scale differences between features using the formula $\mathbf{X}_{\text{std}} = (\mathbf{X} - \mu)/\sigma$, where μ is the mean and σ is the standard deviation. The second step is to compute the covariance matrix $\mathbf{C}$ of $\mathbf{X}_{\text{std}}$ to capture the linear relationships between features: $\mathbf{C} = (\mathbf{X}_{\text{std}}^T \mathbf{X}_{\text{std}})/(n-1)$, where n is the number of samples. The third step is to perform eigen decomposition on $\mathbf{C}$, which means solving the characteristic equation $\det(\mathbf{C} - \lambda \mathbf{I}) = \mathbf{0}$ to obtain eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$. For each eigenvalue λ_i, we need to solve $(\mathbf{C} - \lambda_i \mathbf{I})\mathbf{v}_i = \mathbf{0}$ to find the corresponding eigenvector $\mathbf{v}_i$, and normalize it to a unit length. The so-called "retaining proportion z of the variance" is sorting the eigenvalues in descending order to calculate the cumulative explained variance ratio: $\sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{m} \lambda_i$, and choosing the first k principal components that explain at least z of the total variance. The last step is to construct the projection matrix $\mathbf{P}$ using the selected k eigenvectors and project the original data onto this new k-dimensional space with $\mathbf{X}_{\text{PCA}} = \mathbf{X}_{\text{std}} \cdot \mathbf{P}$. The PCA effectively reduces data dimensions while preserving essential information, which is useful for data compression, feature extraction, and noise reduction. The choice of proportion z depends on balancing information retention and dimensionality reduction efficiency for the specific application. In the ML tasks, we use $\mathbf{X}_{\text{PCA}}$ to instead the original feature matrix $\mathbf{X}$. In our work, we set $z = 1$. This is because when performing PCA on the feature matrix of small molecules, retaining 100% of the variance is crucial due to the limited number of topological features, each of which may be significant. This means that the true result of PCA is to remove the redundant zero columns, making X a full-rank matrix. This ensures no critical information is lost, allowing for precise classification and prediction.

2 Matlab Code

```matlab
X = data{:,1:end-1};
rt = 1;
[coeff, score, ~, ~, explained] = pca(X);
numComponents = find(cumsum(explained) >= 100 * rt, 1);

if ~isempty(numComponents)
    X_pca = score(:, 1:numComponents)';
else
    rt = 0.9999;
    numComponents = find(cumsum(explained) >= 100 * rt, 1);
    X_pca = score(:, 1:numComponents)';
end
```

The MATLAB code above performs PCA on a feature matrix $\mathbf{X}$, selecting the number of components based on the variance explained ratio z. Note that the variable `rt` in the code corresponds to the ratio z discussed in the text.

3 Code Comments

In most cases, setting `rt` $= 1$ can significantly reduce the number of predictors while maintaining 100% variance explanation. However, in some rare cases, due to limitations in machine precision, it is not possible to achieve 100% variance explanation. In such situations, the resulting $\mathbf{X}_{\text{PCA}}$ is an empty matrix. Therefore, we added a condition to the code: if $\mathbf{X}_{\text{PCA}}$ is empty when using `rt` $= 1$, we reset `rt` to 0.9999 and perform PCA again using this new setting, then output $\mathbf{X}_{\text{PCA}}$.