INF554 Machine and Deep Learning
Lab 4: Dimensionality Reduction and Feature Selection

1 Feature Selection using χ^2 Measure

We will study the effects of *feature selection* techniques in learning tasks. In this exercise, we will implement and apply the χ^2 measure to examine its performance in a classification task. We have a dataset $\mathcal{D}$, features $\{X_j\}_{j=1}^{D}$, where each X_j takes values in $\mathcal{X}_j = \{1, \ldots, V\}$, and label Y takes classes in $C = \{1, \ldots, K\}$. We want an importance score for each attribute X_j. A very well known measure for feature selection is the χ^2 ***measure***:

$$\boxed{\chi^2(X_j, Y) = \sum_{v \in \mathcal{X}_j} \sum_{c \in C} \frac{(O_{vc} - E_{vc})^2}{E_{vc}}} \tag{1}$$

with

$$O_{vc} = \sum_{i=1}^{N} \left(\mathbb{1}(x_j^{(i)} = v) \cdot \mathbb{1}(y_i = c) \right) \approx N \cdot P(X_j = v, Y = c)$$

and

$$E_{vc} = N \cdot \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(x_j^{(i)} = v) \cdot \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y_i = c) \approx N \cdot P(X_j = v) \cdot P(Y = c)$$

where $\mathbb{1}(A)$ is the indicator function that returns 1 iff condition A holds. Recall that in the case of independence we would expect that $P(X, Y) = P(X)P(Y)$. Therefore, you can think of the χ^2 test as a test of independence of random variables. In the case of feature selection, high values for χ^2 imply strong dependence between the feature v and class c which would make v a good feature for classification.

1.1 The data

The dataset (`data.csv`) describes a set of 102 molecules of which 39 are judged by human experts to be musks (class 1) and the remaining 63 molecules are judged to be non-musks (class 2). The goal is to learn to predict whether new molecules will be musks or non-musks. However, the 166 features that describe these molecules depend upon the exact shape, or conformation, of the molecule. Because bonds can rotate, a single molecule can adopt many different shapes. To generate this data set, all the conformations of the molecules were generated to produce

6598 conformations. Then, a feature vector was extracted that describes each conformation. The final dataset has 6598 instances and 166 features.

1.2 Tasks

The provided code loads the data into `X` and extracts the class labels into `Y`. Also, a logistic regression function trains a Logistic Regression classifier using the data that are stored on matrix X and the labels Y. The trained classifier can be used to predict the class of new data instances (i.e., molecules in our case). The data that will be used for prediction are stored in the `test.csv` file.

Note that, the prediction produced by the classifier is not binary (i.e., $0-1$ values), but in the range $[0,1]$. We use 100 different threshold values (from 0 to 1 with step 0.01) to assign class labels (if the prediction is less that the threshold, then we assign the first label). Then, for each case we compute the precision and recall and we plot the corresponding curve (see next paragraph for more details on the evaluation).

In order to compare models, we use the **Receiver Operating Characteristic** curve. This curve plots the true positive rate (TPR) (in y-axis) against the false positive rate (FPR) (in x-axis) at various threshold settings, with:

$$\text{TPR} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \qquad \text{FPR} = \frac{\text{false positve}}{\text{false positive} + \text{true negative}}.$$

The higher the **Area Under the ROC Curve (AUC)**, the most performing the model according to this criteria.

Task 1
Run the logistic regression classifier without feature selection. Display the ROC curve (and the area under the curve), as well as the running time.

Task 2
Implement the χ^2 measure for feature selection.

Task 3
Apply feature selection using the χ^2 measure. For different number of features (e.g., selecting only the $20, 40, 60, 80, 100, 150$ features with highest measures) examine the precision-recall curve (and the area under curve), as well as the running time. Does feature selection improve the accuracy? What about the execution time?

2 Dimensionality Reduction with PCA

Principal components analysis (PCA) is a very popular technique for dimensionality reduction. Given a set of data on n dimensions, PCA aims at finding a linear subspace of dimension d lower than n such that the data points lie mainly on this linear subspace. Such a reduced subspace attempts to maintain most of the variance of the data. The linear subspace can be specified by d orthogonal vectors that form a new coordinate system, called the *principal components*. The principal components are linear transformations of the original data points, so there can be no more than n of them. However, the hope is that only $d < n$ principal components are needed to approximate the space spanned by the n original axes. Thus, for a given set of data vectors $x_i, i \in 1 \ldots t$, the d principal axes are those orthonormal axes onto which the variance retained under projection is maximal. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components.

Recall that the variance of a random variable is given by the following formula: $V(X) = \sigma^2 = E[(X - \mu)^2]$. PCA can be computed using the eigenvalue decomposition of the covariance matrix as follows:

1. Suppose that the data is organized into an $m \times n$ matrix $\mathbf{A}$, initially subtract mean values from columns: $\mathbf{C} = \mathbf{A} - \mathbf{M}$
2. Calculate the covariance matrix $\mathbf{W} = \mathbf{C^T C}$
3. Find eigenvalues and eigenvectors of the covariance matrix $\mathbf{W}$
4. *Principal Components*: the k eigenvectors $\mathbf{U}_{[1\ldots k]}$ that correspond to the k largest eigenvalues
5. Project the data to the new space: $\mathbf{CU}_{[1\ldots k]}$

We compute the principal components using the SVD decomposition. Recall that the square root of the eigenvalues of the covariance matrix $\mathbf{C^T C}$ corresponds to the singular values of $\mathbf{C}$. That way, in step 2, we can compute the SVD decomposition of $\mathbf{C}$, and the principal components will correspond to the singular vectors that correspond to the k largest singular values.

As we have already discussed, PCA transforms the data into a lower dimensional space preserving the variance. The fraction of variance that is preserved in the transformed data can be captured by the following formula:

$$var = \frac{\sum_{i=0}^{k} \lambda_i}{\sum_{i=0}^{n} \lambda_j},$$

where n is the number of dimensions in the original dataset, k is the number of dimensions in the new representation space, and λ are the eigenvalues of the covariance matrix sorted in descending order.

2.1 The Data

Here, we will work on data discribing US cities. This dataset is composed of 329 bservations, describing the quality of life with the usage of 9 parameters (housing, climate, education, etc.). The number of dimensions of the dataset is 9.

2.2 Tasks

Task 4
Implement PCA.

Task 5
Run a PCA on the US cities dataset. Visualize a scatterplot of US cities in the 2-dim space obtained from the first two principal components.

3 Rank Reduction with Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a matrix decomposition method that leads to a low-dimensional representation of a high-dimensional matrix. Let $\mathbf{A}$ be a real $m \times n$ matrix. Then the Singular Value Decomposition of matrix $\mathbf{A}$ is defined as

$$\boxed{\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}}}$$

where

- $\mathbf{U}$: $m \times m$ matrix has as columns the eigenvectors of $\mathbf{A}\mathbf{A}^{\mathbf{T}}$
- Σ : $m \times n$ is a diagonal matrix with the singular values of $\mathbf{A}$ in the diagonal (= square roots of $\mathbf{A}\mathbf{A}^{\mathbf{T}}$ eigenvalues)
- $\mathbf{V}$: $n \times n$ matrix has as columns the eigenvectors of $\mathbf{A}^{\mathbf{T}}\mathbf{A}$.

SVD allows an exact representation of any matrix, and also makes it easy to eliminate the less important parts of that representation in order to produce an approximate representation with any desired number of dimensions – leading to the concept of *low rank approximation* of a matrix. Let $\mathbf{A}$ be a $m \times n$ matrix, where $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}}$. Then, a r rank approximation of $\mathbf{A}$ is given by

$$\mathbf{Y} = \mathbf{U}_{m\times r}\mathrm{diag}(\sigma_1, \dots, \sigma_r)\mathbf{V^T}_{r\times n}.$$

The quality of low rank approximation $\mathbf{Y}$ of a matrix $\mathbf{A}$ can be evaluated using the *Frobenious norm* $\| \mathbf{A} - \mathbf{Y} \|_{\mathbf{F}}$, where

$$\| \mathbf{X} \|_{\mathbf{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |x_{ij}|^2}$$

3.1 Tasks

Task 6
Implement SVD on the `gatlin.csv` image. Reconstruct the original matrix (image) $\mathbf{X}$ using the top (largest) $k = \{10, 20, 50, 100, 200\}$ singular values. Compare (visually) these approximations to the original image.

Task 7
What is the error of each approximation? What do you observe? Examine the distribution of the singular values of the original matrix.

4 (Bonus) Multidimensional Scaling (MDS)

Multidimensional Scaling (MDS) is a dimensionality reduction technique used in information visualization, and in particular as a way to display the information contained in a distance matrix. The goal of the method is to place each object (point) in the N-dimensional space such that the between-object distances are preserved as well as possible. Next we give the steps of the MDS algorithm:

1. Let $\mathbf{A}$ be the distance matrix. Construct the $N \times N$ squared distance matrix $\mathbf{D} = \mathbf{A}^2$
2. Compute the matrix $\mathbf{J} = \mathbf{I}_N - N^{-1}\mathbf{1}\mathbf{1}^T$ (where $\mathbf{I}_N$ the $N \times N$ identity matrix (`ones` in diagonal and zeros elsewhere) and $\mathbf{1}\mathbf{1}^T$ is the $N \times N$ matrix with `ones` at each cell)
3. Compute matrix $\mathbf{B} = -\frac{1}{2}\mathbf{JDJ}$
4. Compute the SVD decomposition of matrix $\mathbf{B}$: $\mathbf{B} = \mathbf{U\Sigma V^T}$, and extract the k largest singular values $\mathbf{\Sigma}_k = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k)$ and the corresponding singular vectors $\mathbf{U}_k$
5. A k-dimensional spatial configuration of the N objects is derived by: $\mathbf{A}_k = \mathbf{U}_k\mathbf{\Sigma}_k^{1/2}$, where $\mathbf{\Sigma}_k$ the k largest singular values

The basic idea behind MDS is that the data points are moved in the space in such a way that the *stress criterion* is minimized. Stress is given by the following formula:

$$Stress = \sqrt{\frac{\sum_{\forall i,j}(d_{ij} - \delta_{ij})^2}{\sum_{\forall i,j}\delta_{ij}^2}}, \tag{2}$$

where d_{ij} the distance between points i and j after the trasformation and δ_{ij} their distance in the original space.

Task 8
(Bonus) Implement MDS and apply it to compute the position of 10 US cities in the 2-dimensional space, using the 10×10 distance matrix $\mathbf{D}$. What do you observe? Has the relative position of the cities been preserved?