KPCA DESCRIPTION

Student:
Evgeniya Ovchinnikova

February 12, 2018

Kernel Principal Component Analysis (KPCA)

N-grams representation allows a simple calculation of the distance between long sentences. It is not much more complex than a simple keywords approach, but allows to take into consideration sequences of the words, which is important for avoidance of senseless word order similar to the one in the following example [2]: "A complex pattern-classification problem, cast in a **low-dimensional** space nonlinearly, is more likely to be linearly separable than in a **high-dimensional**, provided that the space is not densely populated.". Here words "low-dimensional" and "high-dimensional" are exchanged and a perfectly correct from the keywords point of view formulation of Cover's theorem becomes completely wrong. However, n-gram analysis can help to avoid it, because it is looking not only for "low-dimensional", but for n-grams like "cast in a high-dimensional" and "than in a low-dimensional", so "cast in a low-dimensional" and "than in a high-dimensional" are incorrect.

N-gram distance solely is not enough to create a sensible feature for the machine learning. It requires further preprocessing. [1] describes KPCA for word embedding based on n-gram similarity, where n-grams are sequences of the letters inside the word. In combination with k-means this algorithm was shown to be more effective for german verbs classification [1] than word2vec, because word2vec doesn't take into account morphological structure of the words. That is why this method was chosen to be extended in such a way that it will be accepting sentences and calculate distances between n-grams those are words sequences.

The idea of the KPCA is mapping of data vectors to the feature space for principal component extraction. It is quite similar to the regular PCA, but the inner product in the feature space is computed using a kernel function, which is usually RBF or polynomial. The following derivation is based on [1] and [3]. To perform KPCA one should take the m-dimensional data matrix: $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}]$ and start with PCA representation:

$$\boldsymbol{C}\boldsymbol{v} = \lambda \boldsymbol{v}, \tag{1}$$

where $\boldsymbol{C} = \frac{1}{n}\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}$ – covariance matrix of $\boldsymbol{X}$, λ and $\boldsymbol{v}$ – eigenvalue and corresponding eigenvector. That is why 1 can be rewritten in the following way:

$$\frac{1}{n\lambda}\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{v} = \boldsymbol{X}\boldsymbol{\alpha} = \boldsymbol{v}, \tag{2}$$

where $\boldsymbol{\alpha} \in R^m$ and it will play a role of an eigenvector for kernalized PCA:

$$\boldsymbol{X}\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{v} = n\lambda\boldsymbol{X}\boldsymbol{\alpha}, \tag{3}$$

adding $\boldsymbol{X}^{\boldsymbol{T}}$ from the left:

$$\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{v} = n\lambda\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{\alpha}, \tag{4}$$

$\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$ would be the kernel matrix $\boldsymbol{K}$ and $n\lambda = \widetilde{\lambda}$, which would be the new eigenvalue:

$$\boldsymbol{K}\boldsymbol{K}\boldsymbol{v} = \widetilde{\lambda}\boldsymbol{K}\boldsymbol{\alpha}, \tag{5}$$

dividing both parts by $\boldsymbol{K}$ we obtain the final form of KPCA equation:

$$\boldsymbol{K}\boldsymbol{v} = \widetilde{\lambda}\boldsymbol{\alpha}, \tag{6}$$

Kernel matrix $\boldsymbol{K}$ is obtained by applying a kernel function to similarity matrix $\boldsymbol{S}$, which is calculated by evaluating similarity between each pair of data elements (words, sentences, etc.) in the given dataset. In our case it is based on Sørensen-Dice or Jaccard distances between the n-grams in sentences.

Bibliography

[1] Brito, Eduardo & Sifa, Rafet & Bauckhage, Christian. (2017). KPCA Embeddings: an Unsupervised Approach to Learn Vector Representations of Finite Domain Sequences.

[2] Cover, T.M., Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition, 1965 (hghlighted words are exchanged)

[3] `https://www.researchgate.net/project/reading-group-machine-learning-AI`