Marta Korpacz

Classnote: Statistical testing in latent variable models

A **latent variable model (LVM)** is a statistical model that takes into account observed (manifest) and unobserved (latent) variables. The variables can be continuous or categorical. We assume the Local Independence, meaning all observed variables are independent if the latent variables are fixed.

The LVMs are useful because they could be used to generate new data and give insight into the low-dimension representation of the high-dimensional data.

There are a lot of reasons why we want low-dimension representation:

- not all variables are important, so we don't want the model to capture the noise or irrelevant patterns, which can lead to overfitting (the situation where our model fits to close to the current dataset like training set and therefore is unable to fir to additional data/make correct predictions on test set).
- high dimensional data requires high computational efficiency and training the model is more time-consuming.
- it is easier to interpret and visualize the model and data if the dimension is lower.

There are two main ways how the researcher can simplify the complex data: feature selection, dimension reduction with techniques such as PCA or t-SNE, or a combination of both. In this classnote, we will focus on feature selection and resampling methods which give the researcher a wide knowledge about data and model parameters.

The goal of the feature selection is to choose the "best" features. There are a lot of different methods and approaches (and also a lot of ways how the researcher can define "best"):

1) **Filter methods**
   In this approach, some of the features are filtered out before the training. The filtering is based on statistical test outcomes. This approach is independent of the LVM we want to use and reduces the risk of overfitting.
   - **Univariate**
     These methods are pretty fast but are not able to capture feature dependencies.
     - ANOVA (Analysis of the Variance), which uses the F-statistic in case of classification problems.

       $$F = \frac{\sum_{i=1}^{K} n_i(\hat{Y}_i - \hat{Y})^2 / (K-1)}{\sum_{i=1}^{K} (Y_{ij} - \hat{Y}_i)^2 / (N-K)},$$ where n_i is the number of observations in the i-th group, $\hat{Y}$ denotes the overall mean of the data, K the number of groups, N the overall sample size, $\hat{Y}_i$ the sample mean in the i-th group, Y_{ij} the j-th observation in the i-th out of K groups.

       In simpler words F is the variance of the group means (Mean Square Between) divided by mean of the within group variances (MSE). Higher F suggests that the feature discriminate well between the classes.

Based on the *F*s p-values are calculated. In this case null hypothesis is that for the selected features there is no difference in means among the groups. The feature selection can be then performed based on some p-value threshold or simply selecting the chosen number of features with highest *F*.
In R it can be performed using aov() from stats. In Python with f_classif from sklearn.feature_selection.

- Kruskal, which uses Kruskal-Wallis rank-sum test:
$H = (N-1)\frac{\sum_{i=1}^{K} n_i(\hat{r}_i - \hat{r})^2}{\sum_{i=1}^{K}\sum_{j=1}^{n_i}(r_{ij} - \hat{r})^2}$, where N is the total number of observations, K is the number of groups, n_i is the number of observation in *i*-th group, $\hat{r}$ is the average od all r_{ij}, where r_{ij} is the rank of observation *j* form group *i* among all observations and $\hat{r}_i$ is an average of ranks in group *i*. It's analogical to ANOVA.
In R *kruskal.test()* from *stats.* In Python *scipy.stats.kruskal()*
- other tests that can be used for feature selection are for example Chi-squared test, Mann-Whitney U test or Pearson correlation.

- **Multivariate**
Multivariate methods are slower than univariate but are capable of capturing feature dependencies because they take into account also the combinations of features.
  - Maximum Relevance Minimum Redundancy (MRMR) uses a greedy iterative approach. In each iteration, the algorithm selects one feature with minimum redundancy to features chosen in previous iterations and maximum relevance to the variable we want to explain.
  In R there is a package *mRMRe* which is able to compute MRMR in parallel: https://cran.r-project.org/web/packages/mRMRe/mRMRe.pdf
  In Python you can use *Pymrmre*: https://pypi.org/project/pymrmre/
  - Relief-based algorithms are a large group of algorithms based on an approach proposed in 1992 by Kira and Rendell. Originally it was designed for feature selection for binary classification tasks, but then generalized for multiclass ones.
  The method helps to find the features in which the examples from the same class are closer in terms of Euclidean distance.
  Step by step the simplest Relief algorithm:
    1. We have a binary target and *n* features and create a vector *w* with zeros in all places and length *n.*
    2. Sample an example *x* from the data.
    3. Find *near-hit*, so an example with the same label as *x,* which is the closest.
    4. Find the *near-miss*, so the closest one with a different label.
    5. Update the *w*:
    $\forall_{F\ in\ features}\ w_F <- w_F - (x_F - nearhit_F)^2 + (x_F - nearmiss_F)^2$
    6. Go back to step 2.

There is an impletantion *relief* in *FSinR* R package (https://search.r-project.org/CRAN/refmans/FSinR/html/relief.html) or in *FSelectorRcpp* (https://search.r-project.org/CRAN/refmans/FSelectorRcpp/html/relief.html).
However the user should take into account that in high-dimensional data the Euclidean distance is high even if all values from two examples are close to each other. In this case, it's better to use more modern implementations such as *VLSReliefF* which has been proved to be efficient even in the case of genome-wide association analysis[11]. In this approach, the distance is calculated for only a subset of the features.

2) **Wrapper methods**

In this case, the feature selection is based on the results of model training. It is slower than other methods and the selection depends on the chosen model, but it can be practical to test the interaction between features and the selected classifier. The disadvantage of wrapper methods is that they can lead to overfitting as they take into account all possible feature combinations, which can make the model too complex. It is good to combine it with other techniques like resampling or regularization like L1 and L2 (more on that later).

- Boruta is an R package proposed by M.B. Kursa and W. R. Rudnicki to find the most relevant variables with *The Boruta Algorithm*. The algorithm is based on Random Forest classification, which output VIM (Variable Important Mesaure).
  Step by step how it works:
  1. Duplicate attributes (if we have less than 5 features, create still at least 5 copies).
  2. Shuffle duplicates to create so-called *shadow attributes.* This step should remove the correlation with the target attribute.
  3. Merge original attributes with the shadow ones.
  4. Perform Random Forest classification to identify important and unimportant attributes. Boruta package compares measures for attributes with the values obtained for shadow ones.
  5. Remove shadow attributes and attributes that have signitificantly worse importance than the shadow ones. The attributes which have significantly better importance than the shadow attributes will be called Confirmed.
  6. Go back to step 1, if the number of reiterations did not reach the defined number of maximum runs or if not all attributes are Confirmed or removed.

  The authors claims that the solution is pretty fast: one hour per one million (attributes * objects) on one core of modern CPU.
  After using Boruta's algorithm, it may also be useful to check different performance of different LVMs to achieve the best possible results (hybrid method)[9].
  A CRAN linking: https://CRAN.R-project.org/package=Boruta

There is also a Python version based on *scikit-learn*: https://pypi.org/project/Boruta/

- backward elimination is an intuitive approach in which we start to train a chosen model with all features and examine it performance after removing the subset of them. The method requires many repetitions of the training process and can be time-consuming, so it can be used only for problems with a small set of features.
  Step by step:
  1. Train a model using all features and evaluate its performance with chosen metrics (accuracy, recall, etc,).
  2. Remove one of the features and train the model again with the remaining data. Evaluate the performance.
  3. Compare the results with those obtained in step 1.
  4. If the performance of the model from step 1. is better than the one from step 2., add the feature back to the dataset, if not, keep it removed.
  5. Repeat the steps.
- Sequential Forward Selection in which we start with an empty set and in each step we increase our set by one feature. In each iteration, we train the model and evaluate the performance to see, in which case evaluation is better. The subset that led to the best performance is then our base set to which we will then use to next increase.

3) **Embedded methods**
   This approach is in most cases something between filter and wrapper methods in terms of time computing efficiency. It uses the learning algorithm in which the feature selection is built in and is performed at the same time as classification or regression.
   - Tree-based algorithms like RandomForest, XGBoost etc. can detect feature interactions and can be used to get the feature importance. However, this approach needs high computational efficiency in the case of high-dimensional data, and in the case of random forest may be not able to detect redundant features[12].
     Random Forest in R: https://CRAN.R-project.org/package=randomForest
     Random Forest in Python: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
   - Penalized methods like LASSO (Least Absolute Shrinkage and Selection Operator), so linear regression with L1 regularization and Elastic Net which is a combination between LASSO and Ridge Regression. Penalization favors simpler models, which should decrease the risk of overfitting.
     LASSO is enabled to set the coefficient to 0 (so the feature is not taken into account) and is effective even in the case of high-dimensional datasets.

Moreover, there is a wide set of **resampling methods**, which allows the researcher to estimate the variability of principal components, linear regression fitting, etc. Below you can find a description of some of them:

Marta Korpacz

1) Bootstrap uses random sampling with replacement to create new data subsets to explore the distribution of the statistic without a need to know data distribution. For example, if we want to explore the average weight of dogs worldwide and we have the data for 10.000 individuals, we can create multiple times bootstrap samples to obtain many subsets with 10.000 examples. For each of them, we can calculate the mean. Then obtained values can be used for standard error or confidence intervals calculation.
   This approach is powerful because it enables us to estimate the variability without obtaining new data, which is not always possible or can be costly. However, it is worth remembering that our results depend strongly on the initial data set. For example, if the person providing the data only measured the weight of dogs coming to a veterinary clinic specializing in small breeds and did not inform us about it, our estimate would differ significantly from the true value.
2) Jackstraw is an approach introduced by Neo Christopher Chung and John D. Storey to identify the association between genomic variables and PCs of interest. Genomic variables as the number of observed variables is much bigger than the number of observations are especially challenging for data analysis.
   Step by step:
   1. We have a $m \, x \, n \;\; (m >> n)$ matrix Y with row-wise mean-centered data, where *m* corresponds to the number of manifested variables and n is the number of observations. Apply Singular Values Decomposition (SVD) and focus on the first *r* PCs: V_r^T. *r* can be challenging to choose. One of the methods is *parallel analysis* (which is implemented in package as *permutationPA*), which compares observed percent variance explained for each of the PCs to the one computed from a randomly permutated dataset. It's important to say that the package authors do not encourage the users to blindly usage of this function and they refer to some papers (Anderson (1963), Tracy and Widom (1996), Johnstone (2001)) regarding the challenge of choosing *r.* Moreover if the user is unsure which value they should pick it is suggest to use the higher one.
   2. Calculate *m* F-statistics to measure if the variance of that variable is explained by the selected PCs.
   3. Randomly select and permutate *s* rows from *Y* (*s <<m)* to obtain *Y`.* Permutated rows we will call *synthetic variables.*
   4. Apply SVD on $Y` = U`D`V`^{`T}$ to get $V`_r^T$.
   5. Calculate null F-statistics from the *s* synthetic rows of *Y`* to see how it will look where there is no real association between variables and PCs.
   6. Repeat steps 3-5 *B* times to get *s x B* null F-statistics, which forms an empirical distribution.
   7. Compute p-values by counting how many times the null F-statistics are greater than or equal to the F-statistic from step 2 and divide it by *s x B.*
   8. Based on p-values identify variables significantly associated with the PCs of interest.

   The method can be used not only for PCA but also for ICA (independent component analysis), K-means clustering and Mini Batch K-means, Partitioning Around Medoids (PAM), or Angle-Based Joint and Individual Variation Explained (AJIVE)[13].
   Here you can find examples of usage in continuous and in categorical data:

http://cran.nexr.com/web/packages/jackstraw/vignettes/jackstraw.pdf
A CRAN linking to the package: https://CRAN.R-project.org/package=jackstraw

References:

1) Kursa, M. B., & Rudnicki, W. R. (2010). Feature Selection with the Boruta Package. Journal of Statistical Software, 36(11), 1–13. https://doi.org/10.18637/jss.v036.i11
2) An Introduction to Statistical Learning (https://www.statlearning.com) by James, Witten, Hastie and Tibshirani
3) Neo Christopher Chung, John D. Storey, Statistical significance of variables driving systematic variation in high-dimensional data, *Bioinformatics*, Volume 31, Issue 4, February 2015, Pages 545–554, https://doi.org/10.1093/bioinformatics/btu674
4) Latent variable models and dimension reduction, Lecture 3, 1000-719bMSB, Neo Christopher Chung
5) https://www.statistics.com/glossaries/
6) Basic Ideas and Examples. (2011). *Wiley Series in Probability and Statistics, 1–18.* doi:10.1002/9781119970583.ch1
7) https://cran.r-project.org/web/packages/flashlight/flashlight.pdf
8) https://CRAN.R-project.org/package=MXM
9) Zhang, Shuguang & Khattak, Afaq & Matara, Caroline & Hussain, Arshad & Farooq, Asim. (2022). Hybrid feature selection-based machine learning Classification system for the prediction of injury severity in single and multiple-vehicle accidents. PloS one. 17. e0262941. 10.1371/journal.pone.0262941.
10) https://towardsdatascience.com/feature-selection-for-the-lazy-data-scientist-c31ba9b4ee66
11) Eppstein, M. J., & Haake, P. (2008). *Very large scale ReliefF for genome-wide association analysis. 2008 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology.* doi:10.1109/cibcb.2008.4675767
12) Pudjihartono, N., Fadason, T., Kempa-Liehr, A., & O'Sullivan, J. (2022). A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction*. Frontiers in Bioinformatics, 2.*
13) Xi Yang, Katherine A. Hoadley, Jan Hannig, J.S. Marron, Jackstraw inference for AJIVE data integration, Computational Statistics & Data Analysis (2023), https://doi.org/10.1016/j.csda.2022.107649
14) https://en.wikipedia.org/wiki/F-test

and all other mentioned directly in the text.

More about LVMs, PCA and SVD can be found in previous classnotes: LVM & PCA,