Machine Learning - Lesson no. 11

Chloé-Agathe Azencott

December 2016

Scribes : Rémi Lalanne, Adam Hotait and Devang Thakkar

1 Introduction

In Machine Learning, it appears necessary to reduce the dimensionality of data set. There are various reasons to do so :

— Computational complexity : the goal is to reduce the computational cost in time and space and therefore to avoid data overfitting.

— Interpretability, so that models are easier to interpret by human users.

— To simplify the models and make them more robust since there is less variance.

— Data visualization : the goal is to make this visualization easier for a human, for instance doctors who want to visualize plenty of data of their patients.

— To curb the cost of data acquisition, that can be very high in some cases.

— To keep only relevant and non-redundant attributes, since irrelevant and redundant attributes can prevent the algorithm from being efficient.

Feature selection and feature extraction are two ways of accomplishing dimensionality reduction.

On the one hand, the idea of feature selection is to select $m < p$ features among the p existing features, and to ignore the remaining $(p - m)$ features. There are three ways to proceed :

1. **Filtering approaches** that calculate a feature relevance score thanks to a statistical measure. Low-scoring features are then removed. These methods have a low computational cost and are independent of the classification algorithm but each feature is considered separately, so they ignore features dependencies.
2. **Wrapper approaches** that aim at finding the best set of features for a given predictive model but that can be computationally very expensive.
3. **Embedded approaches** that simultaneously fit a model and learn which features should be included. They are specific to a model.

All these approaches are supervised.

On the other hand, the idea of feature extraction is to project the p features on $m < p$ new dimensions. There are several approaches, such as linear approaches (Principal Components Analysis, Factor Analysis, MultiDimensional Scaling), supervised approaches (Linear Discriminant Analysis), non linear approaches (Isometric feature mapping, Locally Linear Embedding, Autoencoders).

2 Feature selection : subset selection

In a feature subset selection problem, the issue is to select a subset of features, so that all the attention concerns this subset and that the rest can be ignored.

2.1 Wrappers approaches

The goal is to find the subset of features that leads to the best-performing algorithm, knowing that there are 2^p subsets of p features. Indeed, if we consider p features, each feature can be present or absent in the remaining features, so there are two choices for each future, hence 2^p such sets at the end. Therefore, it is almost impossible to search the whole space exhaustively, except in the cases where n is very small, since complexity is $O(2^p)$.

To solve this problem, we can use a greedy approach ("algorithme glouton" in French) : the forward search. We proceed by adding the "best" feature at each step. Let $E(F)$ be the error on held-out validation set of a predictor trained only using the features in F.

— Initially, $F = \emptyset$
— Then, we find a new best feature : $j* = arg\ min_{j\in\{1,\dots,p\}} E(F \cup \{j\})$
— If $E(F) < E(F \cup \{j\})$: the algorithm stops.
— Else : $F \leftarrow F \cup \{j\}$.

The error is therefore minimal and the complexity is $O(p^2)$, which is way better than $O(2^p)$.

2.2 Embedded approaches

We have seen algorithms which simultaneously optimize a cost/loss function and the features that should be uses. Among them, the $L1$ regularization, with an approach called the Lasso. For a parameter $\lambda \geq 0$, this approach finds $\widehat{\beta}_{lasso} = arg\ min_\beta \|y - X\beta\|_2^2 + \lambda \|\beta\|_2$.

3 Feature extraction : Principal Components Analysis

Another way of reducing the dimensionality of the data is use to Principal Component Analysis. The primary objective that we need to achieve is the minimization of information loss after the data has been projected onto a lower dimensional space using a linear mapping. In other words, we need to find a lower dimensional space, where the variance of the projection is maximised.

A few things to be noted here are :
- PCA is an unsupervised method, in the sense that when project our values, we only consider the data points without paying any heed to the labels that pertain to those points.
- In PCA, one of our early assumptions is that the data that we have is centred ; that is to say, the mean of our data is 0.
- PCA is a variance maximising exercise, hence it requires the data to be normalized before we can apply it correctly.

3.1 Finding the Principal Components

The goal is to find a space which maximizes the variance of the data that is projected on it. Before we begin, we define a certain terms that would enable us to understand better the derivation process. Let w be a unit vector on which we wish to project our matrix of data points X. This would imply that the projection of X in the direction w is given by $z = w^T X$. The variance of z, as calculated in the slides, comes out to be $w^T(\Sigma)w$ where Σ is the covariance of X under the assumptions of a centred dataset.

3.1.1 Finding the first component

We need to begin by finding a vector that maximizes the variance of the projection z and has unit magnitude. A problem of this type takes the form of a Lagrangian optimisation that can be solved easily by introducing a Lagrangian multiplier. Putting it mathematically, the problem is transformed from (1) to (2) :

$$w_1 = \arg\max_{w \in R_p} Var(w^T x) \tag{1}$$

$$w_1 = \arg\max_{w \in R_p} (w^T \Sigma w - \alpha(w^T w \ - \ 1)) \tag{2}$$

As with any ordinary Lagrangian problem, we derive the resulting equation and set it to 0. Doing this leads us to a couple of astonishing results. We find out that :
- The system has been reduced to an eigen value problem

— The multiplier α is an eigen value of the matrix Σ
— The direction vector w_1 is an eigen vector of the matrix Σ
Using this results, we find out that the desired direction vector is the eigen vector with the largest eigen value.

3.1.2 Finding the remaining components

Once we have the first principal component, we proceed to find the remaining components. The process is really similar to the previous case, except for the fact that we have one more condition on our vector w_i, that is, the new vector w_i must be orthogonal to all the previously found components. This algorithm leads to the discovery that the i^{th} component is the eigen vector with the i^{th} largest eigen value.

Once we have all the principal components of the data, we can choose to keep the top m components and discard the rest. An important variable to decide here is m, i.e., how many principal components do we retain. One way to decide is to plot a scree graph of the variance explained by the components and select as many variables as required as long as the addition of one more components seems useful to the total variance explained.

4 Feature extraction : Autoencoders

A third way to do dimensionality reduction is to use neural networks, with the use of Autoencoders. Autoencoders try to reconstruct its input. However, and that is where it uses neural networks, the interest of autoencoders is the hidden layer(s) it constructs. Autoencoders are a non linear, unsupervised approach to dimensionality reduction.

4.1 General presentation

The goal is to select fewer features than there are in the train set to then reconstruct the train set. This is done on several points of data to find the weights that minimize the reconstruction error.

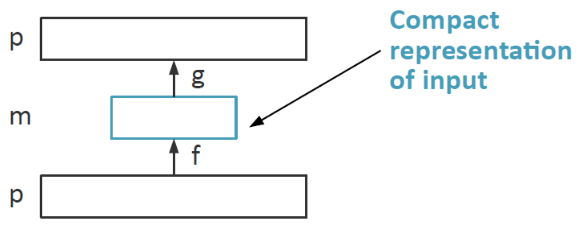

FIGURE 1 – Diagram of a generic autoencoder

4.2 Restricted Boltzmann Machines

RBM are stochastic neural networks, where the activation of a unit is probabilistic, with :

$$P(o_i = 1) = \frac{1}{1 + e^{-a_i + \sum_j o_j w_{ij}}}$$

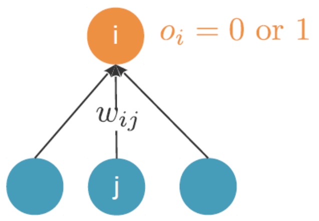

FIGURE 2 – Diagram of a stochastic neural network

A RBM has two layers, with bidirectional connections. The term "restricted" comes from the fact that there are no connection between the hidden units.

The main property of a RBM is the energy of the RBM, defined by :

$$E(\mathbf{x}, \mathbf{z}) = -\sum_{j=1}^{p} a_j x_j - \sum_{h=1}^{H} b_h z_h - \sum_{j=1}^{p}\sum_{h=1}^{H} x_j w_{jh} z_h$$

where a_j is the offset for the visible unit j and b_h is the offset for the hidden unit h.

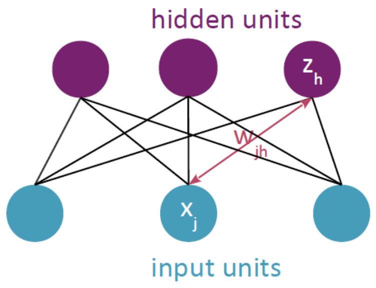

FIGURE 3 – Diagram of a RBM

The goal is to model the probability distribution $P(\mathbf{x})$, with :

$$P(\mathbf{x}) = \frac{\exp(-\mathcal{F}(\mathbf{x}))}{Z}$$

where $\mathcal{F}$ and Z are respectively the free energy and the partition function (which is a normalizing factor), with :

$$\left\{\begin{array}{lcl} \mathcal{F} & = & -\log \sum_{\mathbf{z}} \exp(-E(\mathbf{x},\mathbf{z})) = -\sum_{j=1}^{p} a_j x_j - \sum_{h=1}^{H} \log \sum_{z_h} e^{z_h (b_h + \sum_{j=1}^{p} w_{jh} x_j)} \\ Z & = & \sum_{\mathbf{x}\in\omega} \exp(\mathcal{F}(\mathbf{x})) \end{array}\right.$$

4.3 Deep architectures

It should be noted that by stacking RBM (outputs of RBM become inputs for other subsequent RBM), we get Deep belief networks.

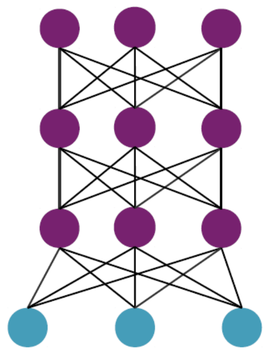

Figure 4 – Diagram of a deep belief network