Dimensionality Reduction

Matthieu R. Bloch

Most machine learning algorithms suffer for what is known as the *curse of dimensionality*. This terminology refers to the problem that generalization often becomes much worse as the dimension of the feature space increases. The exact influence of dimension depends on the specific application and the specific algorithms considered but one trace the cause of problems to the fact that volume increases significantly with dimension. This translates into requiring exponentially more data for the results to be reliable and "cover" a volume of space properly with sample points. When the amount of data is not sufficient in high-dimension, the intuition developed in low dimension can be misleading, as neighboring points (in the sense of being the closest) are not necessarily local anymore. The following example illustrates this point more quantitatively.

Example 0.1. *Consider N points $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ sampled independently uniformly in a p-dimensional unit ball. For every $\mathbf{x}_i \in \mathcal{D}$, let $\ell_i \triangleq \|\mathbf{x}_i\|_2$ be the distance of the i-th point to the unit ball. We are interested in computing the median distance d_{med} from the origin to the closest point in $\mathcal{D}$. Note that for any $d \in \mathbb{R}^+$,*

$$\begin{aligned}\mathbb{P}(\forall i \in [\![1, N]\!]\, \ell_i \geqslant d) &= \prod_{i=1}^{N} \mathbb{P}(\ell_i \geqslant d) && (1)\\ &= \mathbb{P}(\ell_1 \geqslant d)^N && (2)\\ &= (1 - \mathbb{P}(\ell_1 < d))^N && (3)\\ &= (1 - d^p)^N, && (4)\end{aligned}$$

where we have used the fact that the points are sampled independently in (1), identically in (2), and uniformly in (4). By definition of d_{med}, we have

$$\mathbb{P}\left(\min_i \ell_i \geqslant d_{\text{med}}\right) = \mathbb{P}(\forall i \in [\![1, N]\!]\, \ell_i \geqslant d_{\text{med}}) = \frac{1}{2}, \tag{5}$$

so that

$$d_{\text{med}} = \left(1 - \left(\frac{1}{2}\right)^{\frac{1}{N}}\right)^{\frac{1}{p}} \to 1 \text{as } p \to \infty. \tag{6}$$

Therefore, as p gets large, most points are closer to the shell of the unit ball than to the origin. This suggests that the K- nearest neighbors of the origin can actually be quite far.

Dimensionality reduction aims at mitigating the curse of dimensionality by explicitly reducing the dimension of feature vectors. Starting from a dataset of N feature vectors $\mathbf{x}_i \in \mathbb{R}^d$, with d potentially large, the objective is to *transform* inputs to new feature vectors in $\mathbf{x}'_i \in \mathbb{R}^k$ with $k \ll d$ while minimizing the information loss. If successful, dimensionality reduction often improves computational efficiency and helps prevent overfitting, especially when $N \ll d$.

Dimensionality reduction techniques are categorized depending on how they approach the problem. How is information loss measured? Is the approach supervised or unsupervised? Is the map

$\mathbf{x} \to \mathbf{x}'$ linear or non-linear? Most importantly, one distinguishes whether features are *selected*, i.e., only a subset of the components of the original feature vector are retained, or *extracted*, i.e., new components are created. As an illustration compare the vectors

$$\mathbf{x}'_{\text{select}} = \begin{bmatrix} x_1 \\ x_6 \\ x_{32} \end{bmatrix} \quad \text{vs} \quad \mathbf{x}'_{\text{extract}} = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \\ \phi_3(\mathbf{x}) \end{bmatrix}.$$

We will discuss *filtering*, which is a supervised selection method, and *Principal Component Analysis (PCA)*, which is an unsupervised extraction method.

1 Filtering

The objective of filtering is to perform feature selection by eliminating irrelevant features. Features are ranked by order of importance and the best k features are retrained, where the importance is related to the ability of the feature of predicting the associate label y in supervised learning. The objective of this approach is to reduce computational complexity when running learning algorithms, regularize, and retain the interpretability of the feature vectors with reduced dimensions. The main benefit of this approach is that it is reasonably fast, but the k best features is usually not the same thing as the best set of k features.

Ranking for filtering in classification Many possibilities exist to assign a rank $r(j)$ to a feature j. We list here only a few standard ones.

- Misclassification rate: $r(j) \triangleq \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{y_i \neq \theta(x_{i,j})\}$ for some classifier θ_j, which explicitly tries to classify using a single feature.
- Two-sample t-test statistics: $r(j) \triangleq \frac{\left|\overline{x}_j^{(+)} - \overline{x}_j^{(-)}\right|}{s/\sqrt{n}}$ where $\overline{x}_j^{\pm}$ are the class means for feature j and s is the pooled sample standard deviation.
- Margin: for separable data, compute $r(j) \triangleq \min_{k:y_k=+1,\ell:y_\ell=-1} |x_{k,j} - x_{l,j}|$; for non-separable data, one can use order statistics instead.

Ranking for filtering in regression Again, there are many possibilities to rank features. Popular choice include

- Correlation coefficient: $r(j) \triangleq |\rho(j)|$ with

$$\rho(j) \triangleq \frac{\text{Cov}(x_j, y)}{\sqrt{\text{Var}(x_j)\text{Var}(y)}} \tag{7}$$

  where the covariance and variance are computed with sample estimates
- Mutual information: $r(j) \triangleq I(x_j; y)$ where

$$I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{8}$$

  measures how much x tells us about y and the probabilities are estimated from the data.

Ranking is a valid approach but its main drawback is that it can lead to the selection of highly redundant features. For instance, if the best feature were duplicated, it would be selected twice. One solution to circumvent the issue is to resort to *incremental maximization* and choose feature sequentially, keeping in mind those previously selected. For instance, assume that features $x_{j_1}, \cdots, x_{j_{k-1}}$ have been previously selected. The kth feature with incremental selection and mutual information would be chosen to maximize

$$I(x_{j_k}; y) - \beta \sum_{i=1}^{k-1} I(x_{j_k}; x_{j_i}). \tag{9}$$

Intuitively, this procedure attempts to find the best feature that is the least redundant with all previously selected features.

Incremental maximization mitigates the issue of selecting redundant features, but it does not fully address the inability of filtering to capture *interactions* between features. Two solutions exist to go beyond filtering.

- *Wrapper methods*, which measure the performance of *subsets* of features for the learning algorithm combined with a search across all subsets. Such methods capture the interactions of features but can be painfully slow. Common examples include forward selection, backward elimination.
- *Embedded methods*, which use learning algorithms that incorporate a joint feature selection as part of the model fitting. A common example is LASSO regression, which performs ℓ_1 regularization and adds a penalty to non zero coefficients.

2 Principal Component Analysis

Principal Component Analysis (PCA) is a linear unsupervised feature extraction method based on the sum of squared error to reduce dimension from $\mathbb{R}^d$ to $\mathbb{R}^k$ ($k \leqslant d$). The idea is to approximate the data as $\mathbf{x}_i \approx \boldsymbol{\mu} + \mathbf{A}\boldsymbol{\theta}_i$ with $\boldsymbol{\mu} \in \mathbb{R}^d$, $\boldsymbol{\theta}_i \in \mathbb{R}^k$, and $\mathbf{A} \in \mathbb{R}^{d\times k}$ with orthonormal columns.

Definition 2.1. *Principal Component Analysis consists in solving the problem*

$$\underset{\boldsymbol{\mu},\mathbf{A}:\mathbf{A}^\intercal\mathbf{A}=\mathbf{I},\boldsymbol{\theta}_i}{\textit{argmin}} \sum_{i=1}^{N} \|\mathbf{x}_i - \boldsymbol{\mu} - \mathbf{A}\boldsymbol{\theta}_i\|_2^2 \tag{10}$$

This problem happens to have a closed form solution. The difficult part of this optimization problem is finding $\mathbf{A}$. Given $\mathbf{A}$, it is relatively easy to find $\boldsymbol{\theta}_i$ and $\boldsymbol{\mu}$. We develop the characterization of PCA through a series of three lemmas.

Lemma 2.2. *Assume that $\boldsymbol{\mu}$ and $\mathbf{A}$ are fixed. Then, $\boldsymbol{\theta}_i = \mathbf{A}^\intercal(\mathbf{x}_i - \boldsymbol{\mu})$.*

Proof. For fixed $\boldsymbol{\mu}$ and $\mathbf{A}$, the optimization problem in (10) is a separable least-square problem for which we identified the solution

$$\boldsymbol{\theta}_i = (\mathbf{A}^\intercal\mathbf{A})^{-1}\mathbf{A}^\intercal(\mathbf{x}_i - \boldsymbol{\mu}) = \mathbf{A}^\intercal(\mathbf{x}_i - \boldsymbol{\mu}), \tag{11}$$

since the columns of $\mathbf{A}$ are orthonormal and $\mathbf{A}^\intercal\mathbf{A} = \mathbf{I}$ ■

Lemma 2.3. *Assume $\mathbf{A}$ is fixed and $\boldsymbol{\theta}_i = \mathbf{A}^\intercal(\mathbf{x}_i - \boldsymbol{\mu})$. Then, $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$.*

Proof. Substituting the optimal $\boldsymbol{\theta}$ identified in Lemma 2.3 into (10), we obtain

$$\sum_{i=1}^{N} \|\mathbf{x}_i - \boldsymbol{\mu} - \mathbf{A}\boldsymbol{\theta}_i\|_2^2 = \sum_{i=1}^{N} \|\mathbf{x}_i - \boldsymbol{\mu} - \mathbf{A}\mathbf{A}^\intercal(\mathbf{x}_i - \boldsymbol{\mu})\|_2^2 \tag{12}$$

$$= \sum_{i=1}^{N} \|(\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)(\mathbf{x}_i - \boldsymbol{\mu})\|_2^2 \tag{13}$$

$$= \sum_{i=1}^{N} (\mathbf{x}_i - \boldsymbol{\mu})^\intercal (\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)^\intercal (\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)(\mathbf{x}_i - \boldsymbol{\mu}) \tag{14}$$

$$= \sum_{i=1}^{N} (\mathbf{x}_i - \boldsymbol{\mu})^\intercal (\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)(\mathbf{x}_i - \boldsymbol{\mu}) \tag{15}$$

where we have used the fact that $(\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)$ is a projector since $(\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)^\intercal = (\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)$ and

$$(\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)(\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal) = \mathbf{I} - \mathbf{A}\mathbf{A}^\intercal. \tag{16}$$

Using the stationarity condition with respect to μ for (15), we obtain

$$0 = -2\sum_{i=1}^{N} (\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)(\mathbf{x}_i - \boldsymbol{\mu}) = -2(\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)\left(\sum_{i=1}^{N} \mathbf{x}_i - N\boldsymbol{\mu}\right). \tag{17}$$

One (but not unique) solution is $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$. ∎

Lemma 2.4. *One possible choice of* $\mathbf{A}$ *is* $\mathbf{A} = [\mathbf{u}_1, \cdots, \mathbf{u}_k]$ *where* $\mathbf{u}_i$*'s are the eigenvectors corresponding to the* k *largest eigenvalues of* $\mathbf{S} \triangleq \sum_{i=1}^{N} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\intercal$

Proof. The proof proceeds in three distinct steps. First, we assume without loss of generality that $\boldsymbol{\mu} = \mathbf{0}$ so that the PCA problem after using Lemma 2.2 reduces to

$$\underset{\mathbf{A}:\mathbf{A}^\intercal\mathbf{A}=\mathbf{I}}{\operatorname{argmin}} \sum_{i=1}^{N} \|(\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)\mathbf{x}_i\|_2^2 = \underset{\mathbf{A}:\mathbf{A}^\intercal\mathbf{A}=\mathbf{I}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathbf{x}_i^\intercal (\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)^\intercal (\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)\mathbf{x}_i \tag{18}$$

$$= \underset{\mathbf{A}:\mathbf{A}^\intercal\mathbf{A}=\mathbf{I}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathbf{x}_i^\intercal (\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal)\mathbf{x}_i \tag{19}$$

$$= \underset{\mathbf{A}:\mathbf{A}^\intercal\mathbf{A}=\mathbf{I}}{\operatorname{argmax}} \sum_{i=1}^{N} \mathbf{x}_i^\intercal \mathbf{A}\mathbf{A}^\intercal \mathbf{x}_i, \tag{20}$$

where we have used again the fact that $\mathbf{I} - \mathbf{A}\mathbf{A}^\intercal$ is a projector. Now, since for any $\mathbf{v} \in \mathbb{R}^d$ we have $\|\mathbf{v}\|_2^2 = \operatorname{tr}(\mathbf{v}\mathbf{v}^\intercal)$, we obtain

$$\mathbf{x}_i^\intercal \mathbf{A}\mathbf{A}^\intercal \mathbf{x}_i = \|\mathbf{A}^\intercal \mathbf{x}_i\|_2^2 = \operatorname{tr}(\mathbf{A}^\intercal \mathbf{x}_i \mathbf{x}_i^\intercal \mathbf{A}). \tag{21}$$

Upon defining $\mathbf{S} \triangleq \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^\intercal$, the PCA optimization problem reduces to

$$\underset{\mathbf{A}:\mathbf{A}^\intercal\mathbf{A}=\mathbf{I}}{\operatorname{argmax}} \operatorname{tr}(\mathbf{A}^\intercal \mathbf{S}\mathbf{A}). \tag{22}$$

Next, we rewrite the optimization problem as a linear program. We use the fact that $\mathbf{S}$ is symmetric semidefinite positive and the spectral theorem to write $\mathbf{S} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\intercal$ with $\mathbf{U} \in \mathbb{R}^{d\times d}$ an orthonormal matrix and $\boldsymbol{\Lambda}$ a diagonal matrix with elements $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_d \geqslant 0$. Then,

$$\text{tr}\left(\mathbf{A}^\intercal\mathbf{S}\mathbf{A}\right) = \text{tr}\left(\mathbf{A}^\intercal\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\intercal\mathbf{A}\right) = \text{tr}\left(\mathbf{W}^\intercal\boldsymbol{\Lambda}\mathbf{W}\right), \tag{23}$$

where we have defined $\mathbf{W} = \mathbf{U}^\intercal\mathbf{A} \in \mathbb{R}^{d\times k}$. Note that $\mathbf{W}^\intercal\mathbf{W} = \mathbf{A}^\intercal\mathbf{U}\mathbf{U}^\intercal\mathbf{A} = \mathbf{I}$, so that the columns of $\mathbf{W}$ are orthonormal. Upon writing $\mathbf{W} = [w_{ij}]_{i\in[\![1,d]\!],j\in[\![1,k]\!]}$ explicitly in terms of its k orthonormal column vectors in $\mathbb{R}^d$ as

$$\mathbf{W} = \begin{bmatrix} | & | & & | \\ \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_k \\ | & | & & | \end{bmatrix}, \tag{24}$$

we obtain

$$\text{tr}\left(\mathbf{W}^\intercal\boldsymbol{\Lambda}\mathbf{W}\right) = \sum_{j=1}^{k}\mathbf{w}_j^\intercal\boldsymbol{\Lambda}\mathbf{w}_j = \sum_{j=1}^{k}\sum_{i=1}^{d} w_{ij}^2\lambda_i \triangleq \sum_{i=1}^{d} h_i\lambda_i \tag{25}$$

where we have defined $h_i \triangleq \sum_{j=1}^{k} w_{ij}^2$. Note that, by definition $h_i \geqslant 0$ for $i \in [\![1,d]\!]$. In addition,

$$\sum_{i=1}^{d} h_i = \sum_{j=1}^{k}\sum_{i=1}^{d} w_{ij}^2 = \sum_{j=1}^{k}\mathbf{w}_j^\intercal\mathbf{w}_j = k \tag{26}$$

since $\|\mathbf{w}_j\|_2 = 1$. Finally we note that we can augment the matrix $\mathbf{W}$ with $d-k$ orthonormal vectors to form an orthonormal basis of $\mathbb{R}^d$ and write

$$\mathbf{V} = \begin{bmatrix} \mathbf{W} & \mathbf{W}_0 \end{bmatrix} \text{ with } \mathbf{W}_0 \in \mathbb{R}^{d\times d-k} \text{ s.t. } \mathbf{W}_0^\intercal\mathbf{W}_0 = \mathbf{I}_{d-k} \text{ and } \mathbf{W}^\intercal\mathbf{W}_0 = \mathbf{0}. \tag{27}$$

Since $\mathbf{V}$ is then an orthonormal matrix, we also have $\mathbf{V}\mathbf{V}^\intercal = \mathbf{I}$ so that Consequently,

$$h_i = \sum_{j=1}^{k} w_{i,j}^2 \leqslant \sum_{j=1}^{d} v_{i,j}^2 = 1. \tag{28}$$

Therefore, our PCA optimization problem in (10) is equivalent to the linear program

$$\max_{h_i\in[0,1]:\sum_{i=1}^d h_i=k}\sum_{i=1}^{d} h_i\lambda_i. \tag{29}$$

Finally, we solve this linear program explicitly. Recall that the eigenvalues satisfy $\lambda_1 \geqslant \cdots \geqslant \lambda_d$, so that the solution of the linear program consists in allocating the highest possible coefficient in front of the highest eigenvalues. In other words,

$$h_i = \begin{cases} 1 \text{ if } i \in [\![1,k]\!] \\ 0 \text{ otherwise} \end{cases} \tag{30}$$

Ultimately, we are interested in a matrix $\mathbf{A}$. Since $h_i \triangleq \sum_{j=1}^{k} w_{ij}^2$, one possible choice of $\mathbf{W}$ leading to optimal coefficients h_i's is

$$\mathbf{W} = \begin{bmatrix} \mathbf{I}_k \\ \mathbf{0} \end{bmatrix} \text{ leading to } \mathbf{A} = \mathbf{U}\mathbf{W} = \begin{bmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_k \\ | & & | \end{bmatrix}. \tag{31}$$

■