Dimensionality Reduction

Alessandro Leite

November 15th, 2019

Outline

Introduction

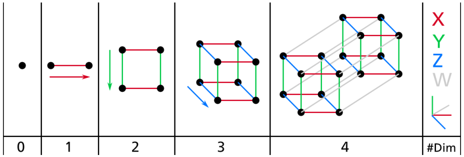
- Machine learning problems may involve thousands or even millions of features for each training set
- Model training in this case may require considerable computing time
- It can also be hard to find a good solution
- This kind of problem characterizes the **curse of dimensionality**
- It is usually possible to reduce the number of features
- **Dimensionality reduction** accelerates training time and it is useful for data visualization
- Reducing the number of dimensions down to two or three makes possible to plot a high-dimensional training set and to visually detect patterns

What is dimensionality reduction?

Dimensionality reduction

It is the process of taking data in a **high dimensional** space and mapping them into a **new space** whose dimensionality is much smaller.

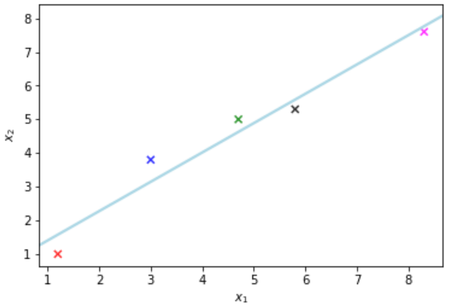
The curse of dimensionality

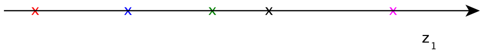


- Human intuition normally fails when trying to image a high-dimensional space
- Even a 4D hypercube may be hard to picture in human mind
- The more dimensions a training set has, the greater the risk of overfit
- One solution to the curse of dimensionality comprises in increasing the size of the training set to reach a sufficient density of training samples
- In practice, the number of training samples required to reach a certain density grows exponentially with the number of dimensions
- For example, a data set with 100 features, would require more training samples that the number of atoms in the observable universe for training samples to be within 0.1 each of other on average, assuming they were spread out uniformly across all dimensions

What are the reasons to reduce data dimensionality (1/4)?

Data compression





- **Reduce data** from 2D to 1D

$$\begin{aligned}
x^1 &\in \mathbb{R} \mapsto z^1 \in \mathbb{R} \\
x^2 &\in \mathbb{R} \mapsto z^2 \in \mathbb{R} \\
&\vdots \\
x^n &\in \mathbb{R} \mapsto z^n \in \mathbb{R}
\end{aligned}$$

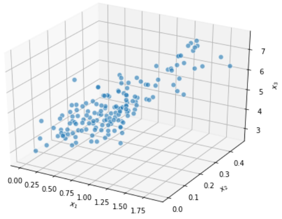
What are the reasons to reduce data dimensionality (2/4)?

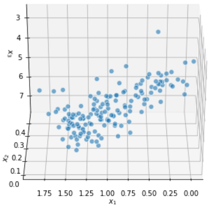
Table 1: World Happiness Report

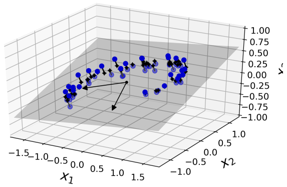
CR	HR	HS	WH	WL	GDP	FM	LE	FR	GE	TGC	DY
Norway	1	7.5	7.59	7.47	1.61	1.53	0.79	0.63	0.36	0.31	2.27
Denmark	2	7.52	7.58	7.46	1.48	1.55	0.79	0.62	0.35	0.40	2.31
Iceland	3	7.50	7.62	7.38	1.48	1.61	0.83	0.62	0.475	0.15	2.32
Switzerland	4	7.494	7.56	7.42	1.56	1.51	0.85	0.62	0.29	0.36	2.27
Finland	5	7.46	7.52	7.41	1.44	1.54	0.80	0.61	0.24	0.38	2.43
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

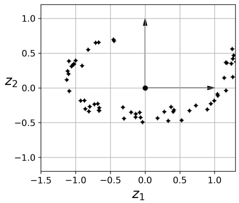
What are the reasons to reduce data dimensionality (3/4)?

Data visualization









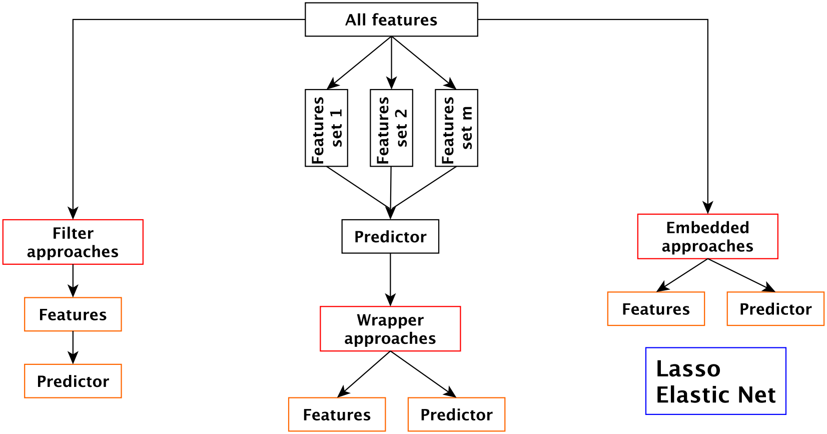
What are the reasons to reduce data dimensionality (4/4)?

- Dimensionality reduction can be used to:
  - reduce storage and computing time
  - help us on understanding a model (e.g., **interpretability**)
  - find meaningful structure of the data
  - visualize the data (e.g., in 2 or 3 dimensions)
  - **remove irrelevant features** that can lead a model to have difficulty in learning
  - reduce the cost of **data acquisition**

There are different strategies to reduce data dimensionality

- **Feature selection**: select m features $m < p$, ignoring the remaining ones
- **Approaches**:
  - **Filtering**: applies a statistical measure to assign a score to each feature (e.g., correlation, χ^2-test)
  - **subset selection**: finds the best set of features for a specific predictive model
  - **Embedded**: simultaneously fits a model and learn which features should be included

Overview of feature selection strategies



Subset selection:

forward selection
backward selection
floating selection

Subset selection

- It aims to find the subset of features that leads to the best-performing model
- Therefore, a brute force strategy needs to deal with 2^p subsets
- We can embrace a **forward search** strategy
  - at each step, we add the best feature to train a predictor
- Given a dataset $\mathcal{D} = (\mathcal{X}, \hat{y})$, where $\mathcal{X} \in \mathbb{R}^{n,p}$, a subset of variables $\varepsilon \subset \{1, \ldots, p\}$, and a $\mathcal{E}(\mathcal{F})$ the error of a predictor trained only using the features in $\mathcal{F}$.

Subset selection

- **Forward search** algorithm
  1. $\mathcal{F} \leftarrow \emptyset$
  2. Find new best feature to include in $\mathcal{F}$:
  $$j^* = \underset{j \in \{1,\ldots,p\}}{\operatorname{argmin}} \mathcal{E}(\mathcal{F} \cup \{j\})$$
  3. stop if $\mathcal{E}(\mathcal{F}) < \mathcal{E}(\mathcal{F} \cup \{j\})$
  4. else $\mathcal{F} \leftarrow \mathcal{F} \cup \{j\}$; go to step 2;
- **What is the complexity?**
  - In the worst case ($\mathcal{F} = \{1, \ldots, p\}$), it's $\mathcal{O}(p^2)$
- Other alternative strategies include:
  - **Backward search**: starting from $\{1, \ldots, p\}$, eliminate the feature $\mathcal{E}(\mathcal{F} \setminus \{j\}) \geq \mathcal{E}(\mathcal{F})$
  - **Floating search**: add q features and remove r features

Feature extraction

- Project p features on $m < p$ new dimensions
- There are different methods for linear and non-linear problems, and most of them are **unsupervised methods**
- Linear methods
  - Principal Component Analysis (PCA)
  - Factor Analysis (FA)
  - Non-negative Matrix Factorization (NMF)
  - Linear Discriminant Analysis (LDA)
- Non-linear methods
  - Multidimensional scaling (MDS)
  - Isometric feature mapping (Isomap)
  - Locally Linear Embedding (LLE)
  - Autoencoders

What is principal component analysis?

Principal component analysis (PCA)

- It is a mathematical procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called **principal components**
- The **first principal component** accounts for the maximum variability of the data, and each succeeding component accounts for as much of the remaining variability
- PCA aims to find a low-dimensional space such that **variance** is **maximized** when the data are projected on that space.
- It is an **unsupervised method**, as we look only at the data and not on any label.
- This method requires **feature standardization**

Feature standardization

1. **Variance** of feature j in dataset $\mathcal{D}$, $\mathcal{D} = \{x^1, \ldots, x^p\}$ $x \in \mathbb{R}^{nxp}$

$$\sigma_j^2 = \frac{1}{n}\sum_{i=1}^{n}(x_j^i - \mu_j)^2$$

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_j^i$$

2. Data normalization:
   - **mean centering**: give each feature a mean of 0
   - **variance scaling**: give each feature a variance of 1

$$x_j^i \hookleftarrow \frac{x_j^i - \mu_j}{\sigma_j}$$

PCA algorithm

PCA

Principal components are features constructed as linear combinations of given features. In this case, the **first principal component** is given by the direction of the **maximum variance** in the data. The **second principal component** is the direction of maximum variance orthogonal to the first component, and so on.

Goal

Find a low-dimensional space such that **variance** is **maximized** when the data are projected on that space.

- **Assumption**: data are **centered** (i.e., they have zero mean)
  - When they don't, we have to subtract the mean:

$$X \leftarrow X - \boldsymbol{\mu}$$

PCA algorithm

- We want to project x in the direction of a matrix $\boldsymbol{w}$, $||\boldsymbol{w}|| = 1$

$$z = X\boldsymbol{w}$$

- The dimensions of z, X, and $\boldsymbol{w}$ are: $(n, 1)$, (n, p), and $(p, 1)$, where n is the number of samples, p is the number of features.
- We can compute $\boldsymbol{Var}(z)$ in function of $\boldsymbol{X}$ and w

$$\begin{aligned}
Var(z) &= Var(X\boldsymbol{w}) \\
&= Var(X^T\boldsymbol{w}^T) \\
&= \mathbb{E}[((X^T\boldsymbol{w}^T) - \mathbb{E}[\boldsymbol{w}^T X^T])^2] \\
&= \mathbb{E}[(\boldsymbol{w}^T X^T - \boldsymbol{w}^T \mathbb{E}[X]^2)] \\
&= \mathbb{E}[\boldsymbol{w}^T X^T X\boldsymbol{w}] \\
&= \boldsymbol{w}^T \mathbb{E}[X^T X]\boldsymbol{w}
\end{aligned}$$

- The dimensions are: $(1, p)$ x (p, n) x (n, p) x $(p, 1)$

Computing the principal components: algorithm

- Reducing data from n-dimensions to k-dimensions
  - Compute the covariance matrix Σ

$$\Sigma = \frac{1}{n} \sum_{i=1}^{n} x^i {x^i}^T$$

  - Compute the eigenvectors of matrix Σ

$$U, S, V = svd(\Sigma)$$

Computing the principal components: theory behind

Let $X \in \mathbb{R}^{n_x p}$ be a centered matrix of covariance $\Sigma = \frac{1}{n}X^T X$. The principal components of X are the eigenvectors of Σ, ordered by their decreasing eigenvalues.

- For all vector $\vec{w} \in \mathbb{R}^p$, the variance of the project of $X \mapsto \vec{w}$ is $\boldsymbol{w}^T \Sigma \boldsymbol{w}$
- The projection of $X \in \mathbb{R}^{n_x p}$ onto $\vec{w} \in \mathbb{R}^p$ is the vector $\vec{z}$

$$\vec{z} = X\boldsymbol{w}$$

- X is **centered**. It means that the **mean** of $\vec{z}$ is:

$$\begin{aligned} &= \frac{1}{n}\sum_{i=1}^{n} z_i \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p} x_j^i w_j \\ &= \frac{1}{n}\sum_{j=1}^{p} w_j \sum_{i=1}^{n} x_j^i \\ &= 0 \end{aligned}$$

Computing the principal components: theory behind

- $Var[\vec{z}]$

$$\begin{aligned} Var[\vec{z}] &= \frac{1}{n}\vec{w}^T X^T X \vec{w} \\ &= \vec{w}^T \Sigma \vec{w} \end{aligned}$$

- Let $\vec{w}_1 \in \mathbb{R}^p$ be the first principal component. Thus $\vec{w}_1$ is orthogonal in a way that the variance of $X\vec{w}_1$ is maximal

$$\vec{w}_1 = \underset{\vec{w} \in \mathbb{R}^p}{\operatorname{argmax}}\, \vec{w}^T \Sigma \vec{w}$$
$$\text{subject to} ||\vec{w}_1||_2 = 1$$

- This represents a quadratic optimization problem, under the constraint of $g(\vec{w}) = 0$. In that case, we can solve it introducing the Lagrange multiplier $\alpha_1 > 0$

$$L(\alpha_1, \vec{w}) = \vec{w}^T \Sigma \vec{w} - \alpha_1(||\vec{w}||_2 - 1)$$

- Due to the strong duality, the maximum of $\vec{w}^T \Sigma \vec{w}$ subject to $||\vec{w}||_2 = 1$ is the $\min_{\alpha_1} \sup\limits_{\vec{w} \in \mathbb{R}^p} L(\alpha_1, \vec{w})$. The *supremum* (least upper bound) of Lagrangien is achieved in the point where its gradient is null

$$2\Sigma\vec{w} - 2\alpha_1\vec{w} = 0$$

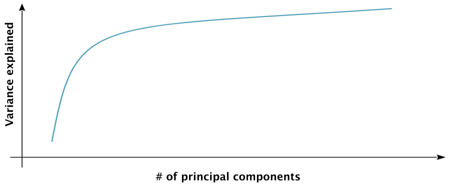
Computing the principal components: theory behind

- As a result, $\Sigma\vec{w_1} = \alpha_1\vec{w}_1$ and $\alpha_1, \vec{w_1}$ are respectively an eigenvalue and an eigenvector of Σ. Considering all the eigenvectors of Σ, $\vec{w}_1$ is the one that maximize the variance

$$\begin{aligned}\vec{w}_1^T\Sigma\vec{w}_1 & \\ &= \alpha_1||\vec{w}_1||_2 \\ &= \alpha_1\end{aligned}$$

How to choose the number of principal components?

- In principal component analysis, we take n dimensional features and reduce them to m feature representation
- Thus, m is a parameter of the PCA algorithm, which is known as the number of principal components
- Choose m in function of the **percentage of variance explained**:
  1. Total variance in the data: $Tr(\Sigma) = \sum_{i=1}^{p} \lambda_i$
  2. The first m principal components accounts for $\frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{p} \lambda_i}$ of the total variance



t-Stochastic Neighbor Embedding (t-SNE)

- It is nowadays a popular method proposed by Maaten and Hinton[1] in 2008
- It approximates the distribution of pairwise distances in the data following a t-distribution[2]

$$\underset{Q}{\operatorname{argmin}} \sum_{i=1}^{n} KL(P_i|Q_i)$$

- where:
  - Q follows a t-distribution
  - KL is the *Kullback-Leibler* divergence (i.e., it measures how much P diverges from Q)
  - P_i is the distribution of the conditional probability that x^i picks x^j as a neighbor. In this case, neighbors are picked in proportion to their probability density under a Gaussian centered in x^i. $P_i = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{||x^j - x^i||^2}{s\sigma^2})$

[1] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605. URL: jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

[2] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. "How to Use t-SNE Effectively". In: *Distill* (2016). URL: distill.pub/2016/misread-tsne.

Multidimensional scaling (MDS)

Goal

Find a mapping that preserves the dissimilarities between the data points.

$$\underset{Z \in \mathbb{R}^{nxm}}{\operatorname{argmin}} \sum_{t=1}^{n} \sum_{u=t+1}^{n} (||z^t - z^u|| - d_{tu})^2$$

- $d_{tu} = ||x^t - x^u||$ In Euclidean space, which is similar to PCA
- Therefore, dissimilarity can also come from other metrics $d : \mathcal{X} x \mathcal{X} \mapsto \mathbb{R}_+$
  - **identity of indiscernible**: $d(x, v) = 0 \Leftrightarrow x = v$
  - **symmetry**: $d(x, v) = d(v, x)$
  - **triangular inequalities**: $d(x, v) \leq d(x, w) + d(w, v)$

References

- Hal Daume III. *A Course in Machine Learning*. 2nd. Self-published, 2017. URL: http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf

  PCA session 15.2
- Max Kuhn and Kjell Johnson. “An Introduction to Feature Selection”. In: *Applied Predictive Modeling*. Springer New York, 2013, pp. 487–519. URL: link.springer.com/chapter/10.1007/978-1-4614-6849-3_19

  Feature selection: from session 19.1 to 19.4
- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd. Springer, 2016. URL: https://web.stanford.edu/~hastie/Papers/ESLII.pdf

  PCA session 14.5.1
  MDS session 14.8
- Isabelle Guyon and André Elisseeff. “An introduction to variable and feature selection”. In: *Journal of Machine Learning Research* 3.Mar (2003), pp. 1157–1182. URL: jmlr.org/papers/v3/guyon03a.html
- Laurens van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605. URL: jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

References

- Martin Wattenberg, Fernanda Viégas, and Ian Johnson. “How to Use t-SNE Effectively”. In: *Distill* (2016). URL: `distill.pub/2016/misread-tsne`
- Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018)
- Jonathon Shlens. “A tutorial on principal component analysis”. In: *arXiv:1404.1100* (2014). URL: `arxiv.org/abs/1404.1100`