

EE445 Mod3-Lec2: Principal Component Analysis & Regression

References:

- [CE-OptMod]: Chapter: 5.3.2
- Additional reference: Chapter 15 of "A Course in ML" by Hal Daumé
(http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf)

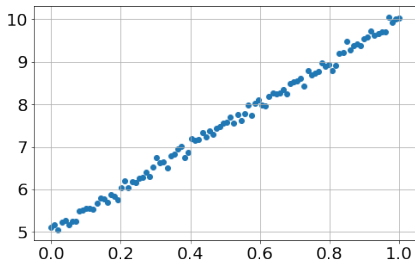
Outline

1. Principal Component Analysis
2. Principal Component Regression

What is PCA?

- Principal component analysis (PCA) is a technique of **unsupervised learning** widely used to “discover” the most important, or informative, directions in a data set.
- Aside **unsupervised learning**: i.e., learning from data without labels or observations—essentially with only features x and no observations y
- There are many reasons you may want to perform **PCA** on a data set
 - ▶ to visualize the data in a lower-dimensional space,
 - ▶ to understand the sources of variability in the data,
 - ▶ to understand correlations between different coordinates of the data points, etc.

What is PCA?



- the majority of the variation of the data is contained in the direction at about 45 degrees from the x -axis
- In contrast, the direction at about 135 degrees contains very little variation.

go to: <https://setosa.io/ev/principal-component-analysis/>

What is PCA? Example

- Suppose we are given dataset $\{x^{(1)}, \dots, x^{(m)}\}$ of attributes of m different types of vehicles, such as their maximum speed, turn radius, and so on.
- Let $x^{(i)} \in \mathbb{R}^n$ with $n \ll m$
- Unknown to us, two different attributes—some x_i and x_j —respectively give a car's
 1. maximum speed measured in miles per hour,
 2. and the maximum speed measured in kilometers per hour.
- These two attributes are therefore almost linearly dependent, up to only small differences introduced by rounding off to the nearest mph or kph
- Thus the data really lies approximately on an $n - 1$ dimensional subspace.
- **How can we automatically detect, and perhaps remove, this redundancy?**

Data Preprocessing: Why?

- It is important to preprocess the data to normalize its mean and variance
- Standardizing the features to have mean zero with a standard deviation of one is important when we compare measurements that have different units.
- Variables that are measured at different scales do not contribute equally to the analysis and might end up creating a bias.

Data Preprocessing: How

Let $(z^{(1)}, \dots, z^{(m)})$ be the original raw data, then preprocessing goes as follows:

Step 1: compute the mean

Step 2: recenter the data

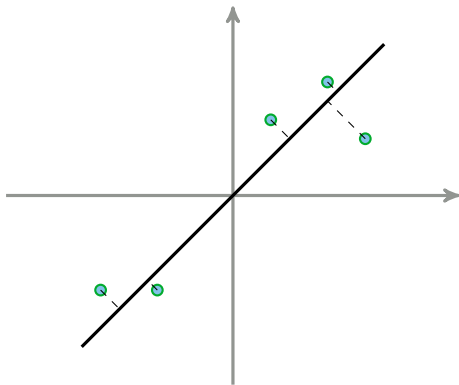
Step 3: compute the standard deviation

Step 4: normalize (scale) the data

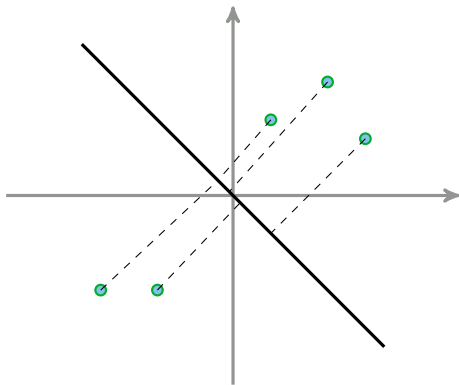
How do we compute the “major axis of variation”?

- We want to compute the direction on which the data approximately lies.
- One way to pose this problem is as finding the unit vector u so that when the data is projected onto the direction corresponding to u , the variance of the projected data is maximized
- In other words, we would like to choose a direction u so that if we were to approximate the data as lying in the direction/subspace corresponding to u , as much as possible of this variance is still retained.

Toy Example



- projected data still has a fairly large variance, and points are far from origin



- projections have a significantly smaller variance, and are closer to the origin

PCA Warm up: Projecting onto first principle component

- Recall: the length of the projection of x onto u is given by $x^\top u$
- To maximize the variance of the projections, we choose a unit-length u to maximize

- **Caution!**: the $x^{(i)}$ here are the pre-processed features—i.e., they are the centered and scaled (normalized) features

PCA Warm up: Projecting onto first principle component

- **How?** This is actually an optimization problem given by

- To solve, we write out the "Lagrangian"

- **Summary:** we have found that if we wish to find a 1-dimensional subspace with which to approximate the data, we should choose u to be the **principal eigenvector** of Σ

What about projecting on to $k = 2$ components?

- To get a second dimension, we want to find a new vector v on which the data has maximal variance, but to avoid redundancy, we want $v^\top u = 0$
- Optimization problem:

- Optimality for the Lagrangian

PCA More Generally

- Suppose we wish to project our data on to a k -dimensional subspace ($k < n$)
- We should choose u_1, \dots, u_k to be the top k eigenvectors of Σ .
- The u_i 's form a new, orthogonal basis for the data
- Indeed, recall that Σ is symmetric so we can always choose the u_i 's to be orthogonal to one another
- Next, we represent each $x^{(i)}$ in the new basis

$$y^{(i)} = (u_1^\top x^{(i)}, u_2^\top x^{(i)}, \dots, u_k^\top x^{(i)}) \in \mathbb{R}^k$$

- $x^{(i)}$ are n -dimensional and $y^{(i)}$ are k -dimensional
- **PCA** is therefore also referred to as a **dimensionality reduction** algorithm.
- Vectors u_1, \dots, u_k are called the first k **principal components**

Summary: PCA Algorithm

- Pre-process the raw data $(z^{(1)}, \dots, z^{(m)})$
 1. Recenter the data: define $\tilde{x}^{(i)} = z^{(i)} - \mu$ where $\mu = \frac{1}{m} \sum_{i=1}^m z^{(i)}$
 2. Rescale/normalize: define $x^{(i)}$ with entries $x_j^{(i)} = \tilde{x}_j^{(i)} / \sigma_j$ where $\sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (\tilde{x}_j^{(i)})^2$
- Run **PCA**
 1. Compute the covariance matrix $\Sigma = \frac{1}{m} \sum_{i=1}^m x^{(i)}(x^{(i)})^\top = \frac{1}{m} X^\top X$
 2. Compute the eigenvalues and (orthonormal) eigenvectors of Σ
 3. Retain k eigenvectors with largest eigenvalues V_k
 4. Project X onto the principal component space

Alternative Derivation via Reconstruction Error

- Rather than maximizing variance, we may want to minimize reconstruction error
- 1-dimensional case: we are looking for a single projection direction u
- projected data: $y = Xu$ where each y_i is the position of the i -th feature vector along u
- To project back into the original space we do $yu^\top = Xuu^\top$ —i.e., yu^\top is the *reconstructed* value
- **Reconstruction Error:**

Connections with SVD

- **Facts:** for a symmetric matrix $\Sigma = \Sigma^\top$,
 - ▶ the singular values are the absolute values of the eigenvalues and $\Sigma = U\Lambda V^\top$ where $U = V$
 - ▶ if $\Sigma \geq 0$, then $\lambda_i \geq 0$
 - ▶ if $\Sigma \succ 0$, then $\lambda_i > 0$ and U, V, Λ are all square non-singular matrices
- Indeed, $\Sigma^\top \Sigma = \Sigma^2$ so that $\sigma_i(\Sigma) = \sqrt{\lambda_i(\Sigma^2)} = \lambda_i(\Sigma)$

Use the **SVD** to scale up!

- Often we have very large data sets—i.e., Σ might be very big in terms of dimension
- **Problems**: Computing eigenvectors is slow, and computing Σ could have numerical precision issues
- As an alternative we can use **SVD** since **PCA** reduces to **SVD**

Reducing PCA to SVD

- $\Sigma = X^T X \in \mathbb{R}^{n \times n}$ is symmetric PSD $\implies \Sigma = Q\Lambda Q^T$ where $QQ^T = I$
- Consider the SVD of $X = USV^T$.

- Hence, the rows of $V^T = Q^T$ are the eigenvectors of $\Sigma = X^T X$
 - ▶ The right singular vectors of X are the same as the eigenvectors of $X^T X$
 - ▶ The eigenvalues of $X^T X$ are the squares of the singular values of X
- Thus PCA reduces to computing the SVD of X (without having to form $X^T X$!).
- Output of PCA is the top k eigenvectors of $X^T X \iff$ SVD of $X = USV^T$ gives top k eigenvectors of $X^T X$ via first k rows of V^T

PCA based Low-Rank Approximations

- The techniques developed for PCA can also be used to produce low-rank matrix approximations.
- We seek matrices Y, Z^T such that $X = YZ^T$

Example: Eigenfaces

Mod3-N3.ipynb