CS 189/289A Introduction to Machine Learning
Fall 2023 Jennifer Listgarten, Jitendra Malik

DIS6

1 Motivation: Dimensionality reduction

In this problem sheet we explore the motivation for general dimensionality reduction in machine learning and derive from first principles why projection on the first eigenvectors of the covariance matrix of the data has some favorable properties. A deeper understanding on the advantages of PCA and other dimensionality reduction methods is conveyed in the homework.

In general, we assume the following scenario: Suppose we are given n points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^d$ and the dimension of the feature vectors is d (very big, like 10^3). By dimensionality reduction, we refer to a mapping $\psi : \mathbb{R}^d \mapsto \mathbb{R}^k$ that maps vectors from $\mathbb{R}^d$ to $\mathbb{R}^k$ with $k \ll d$.

(a) (Motivation) Given n feature vectors of d dimensions, in which regimes of n, d and why would you want to reduce the dimensionality in practical machine learning applications? Think about the concept of regularization studied extensively in the past few weeks.

Solution:

In general:

There are two big motivations for dimensionality reduction. First, there is the simple one coming from the bias/variance tradeoff. You have seen that every feature essentially brings its own variance to the problem that scales like $\frac{1}{n}$. So, when there are a lot of potential features that we could use, the optimal number of features that we do use might be fewer. Hence, dimensionality reduction. The second is computational. Processing the data is costly. If a system of linear equations has to be solved, the complexity is super-linear in the number of variables. Cutting down the number of features cuts down the number of variables that we need to solve for.

Reducing dimension here seems to be win-win. Though for computing the reduction, we need to be smart, else there are no computational savings.

In slightly more detail: There are three basic regimes $n \ll d$ (high dimensional data regime = underdetermined), $n >\sim d$ and $n \gg d$ (overdetermined):

- When $n \ll d$ (underdetermined system), we use regularization (or model selection) assuming low rank structure to avoid the overfitting that would naturally occur.
- When $n \gg d$ there might appear to be enough data to get good estimates of the variables. However, even here, we can potentially get some advantages from dimensionality reduction (and other forms or regularization) if there is lower-dimensional models have good approximation error. (You saw this with polynomials fitting the exponential function.)

Another idea here (not explored that much in 189) is to do "sketching" to reduce the n samples to something that is fewer since there is plenty of data.

- Dimensionality reduction is in general most helpful in cases when $n \ll d$ or $n >\sim d$ and the problem has lower effective dimension (otherwise $n \ll d$ obviously wouldn't give you a reasonable estimator in the first place).

(b) (Computational aspect) Revisit this in the context of linear regression. What is the computational complexity of performing a linear regression of n data points in d dimensions with $n > d$ (say by solving the normal equations when $\mathbf{X}^\top\mathbf{X}$ is invertible)? If the projection was given to you for free, approximately how many operations would you save if you reduced the dimension from $d = 10^3$ to $d = 10$?

Solution:

Solving linear regression requires $O(nd^2)$ for $n > d$ (and $O(n^2d)$ for $n < d$). We discuss the case $n > d$ for which the computational complexity can be seen by considering the solution via normal equations, which is

$$\mathbf{w}^* = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}. \tag{1}$$

Forming the matrix $\mathbf{X}^\top\mathbf{X}$ costs $O(nd^2)$ and inverting it costs $O(d^3)$. Forming $\mathbf{X}^\top\mathbf{y}$ costs $O(nd)$ and the final matrix multiplication of the two $d \times d$ matrices costs $O(d^3)$. The total cost is therefore $O(d^2(n + d))$ which in the case $d < n$ is equal to $O(nd^2)$. An alternative approach to compute the computational complexity of L.R. is via SVD computation.

Reducing the dimension by a factor of 100 therefore gives a 10,000 fold speedup for solving the linear regression, which is quite substantial.

2 Regularized k-Means

Recall that in k-means clustering we attempt to minimize the objective

$$\min_{C_1,C_2,\ldots,C_k} \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|_2^2, \text{ where}$$

$$\mu_i = \text{argmin}_{\mu_i \in \mathbb{R}^d} \sum_{x_j \in C_i} \|x_j - \mu_i\|_2^2 = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j, \quad i = 1, 2, \ldots, k.$$

The samples are $\{x_1, \ldots, x_n\}$, where $x_j \in \mathbb{R}^d$. C_i is the set of sample points assigned to cluster i and $|C_i|$ is its cardinality. Each sample point is assigned to exactly one cluster.

(a) What is the minimum value of the objective when $k = n$ (the number of clusters equals the number of sample points)?

Solution: The value is 0, as every point can have its own cluster.

(b) Suppose we add a regularization term to the above objective. The objective is now

$$\sum_{i=1}^{k} \left(\lambda \|\mu_i\|_2^2 + \sum_{x_j \in C_i} \|x_j - \mu_i\|_2^2 \right).$$

Show that the optimum of

$$\min_{\mu_i \in \mathbb{R}^d} \lambda \|\mu_i\|_2^2 + \sum_{x_j \in C_i} \|x_j - \mu_i\|_2^2$$

is obtained at $\mu_i = \frac{1}{|C_i|+\lambda} \sum_{x_j \in C_i} x_j$.

Solution:

Consider the function

$$f(\mu_i) = \left(\sum_{x_j \in C_i} \|x_j - \mu_i\|_2^2 \right) + \lambda \|\mu_i\|_2^2.$$

Its gradient with respect to μ_i is

$$\begin{aligned} \nabla_{\mu_i} f(\mu_i) &= \left(2 \sum_{x_j \in C_i} (\mu_i - x_j) \right) + 2\lambda \mu_i \\ &= 2 \left((|C_i| + \lambda)\mu_i - \sum_{x_j \in C_i} x_j \right). \end{aligned}$$

Setting it to zero, we have $\mu_i = \frac{1}{|C_i|+\lambda} \sum_{x_j \in C_i} x_j$. As the function f is convex, the minimum is obtained at $\mu_i = \frac{1}{|C_i|+\lambda} \sum_{x_j \in C_i} x_j$.

(c) Here is a simplified example where we would want to regularize clusters. Suppose there are n students who live in a $\mathbb{R}^2$ Euclidean world and who wish to share rides efficiently to Berkeley for their in-person final exam in EECS189/289A. There are k shuttles which may be used for shuttling students to the exam location. The students need to figure out k good locations to meet up. The students will then walk to the closest meet up point and then the shuttles will deliver them to the exam location. Let x_j be the location of student j, and let the exam location be at $(0, 0)$. Assume for simplicity that we can drive by taking the shortest path between two points.

Write down an appropriate objective function to minimize the total distance that the students and vehicles need to travel. How is this different from the regularized k-means objective above?

Solution:

The objective function that minimizes the total distance that the students and vehicles need to travel is

$$\min_{\mu_i \in \mathbb{R}^d} \sum_{i=1}^{k} \left(\|\mu_i\|_2 + \sum_{x_j \in C_i} \|x_j - \mu_i\|_2 \right). \tag{2}$$

It differs from regularized k-means in that there aren't squares on the norms and $\lambda = 1$.

Cathy Wu, Ece Kamar, Eric Horvitz. *Clustering for Set Partitioning with a Case Study in Ridesharing*. IEEE Intelligent Transportation Systems Conference (ITSC), 2016.

3 Kernel PCA

You have seen how to use PCA to do dimensionality reduction by projecting the data to a subspace that captures most of the variability visible in the observed features. The underlying hope is that these directions of variation are also relevant for prediction the quantities of interest.

Standard PCA works well for data that is roughly Gaussian shaped, but many real-world high dimensional datasets have underlying low-dimensional structure that is not well captured by linear subspaces. However, when we lift the raw data into a higher-dimensional feature space by means of a nonlinear transformation, the underlying low-dimensional structure once again can manifest as an approximate subspace. Linear dimensionality reduction can then proceed. As we have seen in class so far, kernels are an alternate way to deal with these kinds of nonlinear patterns without having to explicitly deal with the augmented feature space. This problem asks you to discover how to apply the "kernel trick" to PCA.

Let $\mathbf{X} \in \mathbb{R}^{n \times \ell}$ be the data matrix, where n is the number of samples and ℓ is the dimension of the raw data. Namely, the data matrix contains the data points $\mathbf{x}_j \in \mathbb{R}^{\ell}$ as rows

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} \in \mathbb{R}^{n \times \ell}. \tag{3}$$

(a) Compute $\mathbf{X}\mathbf{X}^\top$ in terms of the singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ where $\mathbf{U} \in \mathbb{R}^{n \times n}, \mathbf{\Sigma} \in \mathbb{R}^{n \times \ell}$ and $\mathbf{V} \in \mathbb{R}^{\ell \times \ell}$. Notice that $\mathbf{X}\mathbf{X}^\top$ is the matrix of pairwise Euclidean inner products for the data points. How would you get $\mathbf{U}$ if you only had access to $\mathbf{X}\mathbf{X}^\top$?

Solution: We have $\mathbf{X}\mathbf{X}^\top = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top$. Here, $\mathbf{\Sigma}^2$ is a $n \times n$ diagonal matrix with zeros on the diagonal as needed. Notice that the columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{X}\mathbf{X}^\top$.

(b) Given a new test point $\mathbf{x}_{test} \in \mathbb{R}^{\ell}$, one central use of PCA is to compute the projection of $\mathbf{x}_{test}$ onto the subspace spanned by the k top singular vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$.

Express the scalar projection $z_j = \mathbf{v}_j^\top \mathbf{x}_{test}$ onto the j-th principal component as a function of the inner products

$$\mathbf{X}\mathbf{x}_{test} = \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}_{test} \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_{test} \rangle \end{pmatrix}. \tag{4}$$

Assume that all diagonal entries of $\mathbf{\Sigma}$ are nonzero and non-increasing: $\sigma_1 \geq \sigma_2 \geq \cdots > 0$.

Hint: Express $\mathbf{V}^\top$ in terms of the singular values $\mathbf{\Sigma}$, the left singular vectors $\mathbf{U}$ and the data matrix $\mathbf{X}$.

Solution: Using the compact form of the SVD, we have $\mathbf{V}^\top = \mathbf{\Sigma}^{-\top}\mathbf{U}^\top\mathbf{X}$. Here, $\mathbf{\Sigma}^{-\top}$ denotes the $l \times n$ matrix with the reciprocal singular values along the main diagonal. Thus,

$$z_j = \mathbf{v}_j^\top \mathbf{x}_{test} = \sigma_j^{-1} \mathbf{u}_j^\top \mathbf{X}\mathbf{x}_{test}.$$

(c) How would you define kernelized PCA for a general kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ (to replace the Euclidean inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$)? For example, the RBF kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\delta^2}\right)$.

Describe this in terms of a procedure which takes as inputs the training data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^\ell$ and the new test point $\mathbf{x}_{test} \in \mathbb{R}^\ell$, and outputs the analog of the previous part's z_j coordinate in the kernelized PCA setting. You should include how to compute $\mathbf{U}$ from the data, as well as how to compute the analog of $\mathbf{X}\mathbf{x}_{test}$ from the previous part.

Solution:

(a) Obtain the vectors $\mathbf{u}_j$ as eigenvectors from the eigendecomposition of $\mathbf{K} \in \mathbb{R}^{n \times n}$ with $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. The eigendecomposition also gives us $\mathbf{\Sigma}^2$ as defined in part (a).

(b) Kernelize the inner products $z_j = \frac{1}{\sigma_j}\mathbf{u}_j^\top \mathbf{X}\mathbf{x}_{test}$ via

$$z_j = \frac{1}{\sigma_j}\mathbf{u}_j^\top \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_{test}) \\ k(\mathbf{x}_2, \mathbf{x}_{test}) \\ \vdots \\ k(\mathbf{x}_n, \mathbf{x}_{test}) \end{pmatrix} \tag{5}$$

.