1 Regularized k-Means

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

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

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

The samples are $\{x_1, \dots, x_n\}$, 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

$$\nabla_{\mu_i} f(\mu_i) = \left(2 \sum_{x_j \in C_i} (\mu_i - x_j) \right) + 2\lambda \mu_i$$

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

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

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

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.

2 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

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} \in \mathbb{R}^{n\times\ell}. \tag{2}$$

(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

$$\mathbf{X}\mathbf{x}_{test} = \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}_{test} \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_{test} \rangle \end{pmatrix}. \tag{3}$$

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

$$z_j = \frac{1}{\sigma_j}\mathbf{u}_j^\top \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_{test}) \\ k(\mathbf{x}_2, \mathbf{x}_{test}) \\ \vdots \\ k(\mathbf{x}_n, \mathbf{x}_{test}) \end{pmatrix} \tag{4}$$

.

3 Theory of Hard-Margin Support Vector Machines

A *decision rule* (or *classifier*) is a function $r : \mathbb{R}^d \to \pm 1$ that maps a feature vector (test point) to $+1$ ("in class") or -1 ("not in class"). The decision rule for linear SVMs is

$$r(x) = \begin{cases} +1 & \text{if } w \cdot x + \alpha \geq 0, \\ -1 & \text{otherwise,} \end{cases} \tag{5}$$

where $w \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ are the weights (parameters) of the SVM. The hard-margin SVM optimization problem (which chooses the weights) is

$$\min_{w,\alpha} \; |w|^2 \;\text{ subject to }\; y_i(X_i \cdot w + \alpha) \geq 1, \;\; \forall i \in \{1, \ldots, m\}, \tag{6}$$

where $|w| = \|w\|_2 = \sqrt{w \cdot w}$.

We can rewrite this optimization problem by using Lagrange multipliers to eliminate the constraints. We thereby obtain the equivalent optimization problem

$$\max_{\lambda_i \geq 0} \min_{w,\alpha} \; |w|^2 - \sum_{i=1}^{m} \lambda_i (y_i(X_i \cdot w + \alpha) - 1). \tag{7}$$

(a) Show that Equation (7) can be rewritten as the *dual optimization problem*

$$\max_{\lambda_i \geq 0} \; \sum_{i=1}^{m} \lambda_i - \frac{1}{4} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j X_i \cdot X_j \;\text{ subject to }\; \sum_{i=1}^{m} \lambda_i y_i = 0. \tag{8}$$

Hint: Use calculus to determine what values of w and α optimize Equation (7). Explain where the new constraint comes from.

SVM software usually solves this dual quadratic program, not the primal quadratic program.

Solution: Taking the gradient with respect to w and α and setting it to zero, we obtain

$$\begin{aligned} w^* &= \frac{1}{2} \sum_{j=1}^{m} \lambda_j y_j X_j, \\ 0 &= \sum_{i=1}^{m} \lambda_i y_i. \end{aligned}$$

The latter equation is our new constraint. Substituting the first equation back into Equation (7) and noting that $\sum_{i=1}^{m} \lambda_i y_i \alpha = 0$, we obtain

$$\begin{aligned} & \max_{\lambda_i \geq 0} \; |w^*|^2 - \textstyle\sum_{i=1}^{m} \lambda_i (y_i X_i \cdot w^* - 1) \\ = \; & \max_{\lambda_i \geq 0} \; \left(\tfrac{1}{2} \textstyle\sum_i \lambda_i y_i X_i\right)^\top \left(\tfrac{1}{2} \textstyle\sum_j \lambda_j y_j X_j\right) - \textstyle\sum_{i=1}^{m} \lambda_i y_i X_i \cdot \left(\tfrac{1}{2} \textstyle\sum_j \lambda_j y_j X_j\right) - \lambda_i \\ = \; & \max_{\lambda_i \geq 0} \; \tfrac{1}{4} \textstyle\sum_i \sum_j \lambda_i \lambda_j y_i y_j X_i \cdot X_j - \tfrac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j X_i \cdot X_j + \sum_i \lambda_i \end{aligned}$$

(b) Suppose we know the values λ_i^* and α^* that optimize Equation (7). Show that the decision rule specified by Equation (5) can be written

$$r(x) = \begin{cases} +1 & \text{if } \alpha^* + \frac{1}{2}\sum_{i=1}^m \lambda_i^* y_i X_i \cdot x \geq 0, \\ -1 & \text{otherwise.} \end{cases} \tag{9}$$

Solution: By substituting the optimal $w^* = \frac{1}{2}\sum_{i=1}^m \lambda_i^* y_i X_i$ into Equation (5), we obtain the specified decision rule.

(c) The training points X_i for which $\lambda_i^* > 0$ are called the support vectors. In practice, we frequently encounter training data sets for which the support vectors are a small minority of the training points, especially when the number of training points is much larger than the number of features (i.e., the dimension of the feature space). Explain why the support vectors are the only training points needed to evaluate the decision rule. What influence do the non-support vectors have on the decision rule?

Solution: Every training point X_i that is not a support vector has $\lambda_i^* = 0$, so X_i makes no contribution to Equation (9). Only the support vectors contribute to Equation (9).

However, every training point that is not a support vector still must be classified correctly by the hard-margin SVM and thus influences the decision rule. Even for soft-margin SVMs, these points will influence the decision rule in terms of either needing to be classified correctly or using up some amount of slack, which incurs a loss.