K-Nearest Neighbors

Statistical Machine Learning (ENGG*6600*08)

School of Engineering,
University of Guelph, ON, Canada

Course Instructor: Benyamin Ghojogh
Fall 2023

k-Nearest Neighbors

k-Nearest Neighbors

- Consider the dataset $\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^n$.
- Some data points are close to each other and some are far from each other, in the d-dimensional Euclidean space. We can calculate the Euclidean distance between every pair of points in the space.
- Consider a point $\boldsymbol{x}_i$. We can calculate the Euclidean distance of all points of dataset, except itself, from this point:

$$\|\boldsymbol{x}_1 - \boldsymbol{x}_i\|_2, \|\boldsymbol{x}_2 - \boldsymbol{x}_i\|_2, \ldots, \|\boldsymbol{x}_n - \boldsymbol{x}_i\|_2.$$

- We sort these distances in ascending order and keep the k smallest distances. The corresponding k points with the smallest distances from $\boldsymbol{x}_i$ are the k-Nearest Neighbors (k-NN) of $\boldsymbol{x}_i$. Note that k is a hyper-parameter positive integer for the number of neighbors.
- We do this procedure for all n points of dataset to have k-NN for all points.
- Consider a graph whose vertices are the indices of the points $\{1, 2, \ldots, n\}$ and an edge from vertex j to vertex i exists if $\boldsymbol{x}_j$ is one of the k nearest neighbors of $\boldsymbol{x}_i$. Such a graph is called the k-**NN graph** of the dataset.

Adjacency Matrix

- Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ whose (i, j)-th element is:

$$\boldsymbol{A}(i,j) := \begin{cases} 1 & \text{if } j \in k\text{-NN}(\boldsymbol{x}_i) \\ 0 & \text{Otherwise.} \end{cases} \tag{1}$$

  This matrix shows the indices of k-NN for all points. It is called the **adjacency matrix** or the k-**NN matrix**.
- Usually, the main diagonal of the adjacency matrix is zero or is ignored because we do not consider every point to be its own neighbor.
- If $k \ll n$, then the adjacency matrix is very sparse having lots of zeros.
- We can have the adjacency matrix between two datasets with size n_1 and n_2. In this case, the size of matrix is $n_1 \times n_2$.
- Note that one may define the adjacency matrix to be $\boldsymbol{A} \in \mathbb{R}^{n \times k}$ where every row lists the indices of the k-NN of $\boldsymbol{x}_i$ in the dataset.

k-NN for Machine Learning

k-NN for Machine Learning

- k-NN can be used for machine learning tasks.
- k-NN for **regression**:
  - For every input point $\boldsymbol{x}$, we find the k-NN of $\boldsymbol{x}$ among the training data points.
  - The average or some statistics of the labels of the k-NN is used as the estimated label of the point $\boldsymbol{x}$.
- k-NN for **classification**:
  - For every input point $\boldsymbol{x}$, we find the k-NN of $\boldsymbol{x}$ among the training data points.
  - The majority of the class labels of the k-NN is used as the estimated class label of the point $\boldsymbol{x}$.
- As explained above, k-NN for regression or classification does not have any training phase but it only has the test phase. The training phase of k-NN in libraries just takes the training data to be used later for k-NN calculation in the test phase.
- The k-NN classifier partitions the space into classes. The smaller the k, the more variance the decision boundary of classes has. Therefore, the small k leads to overfitting.
- In general, larger k in k-NN classifier or regressor results in less overfitting and better generalization. However, larger k has more computation. So, there is a trade-off between generalization and computation.

Kernel k-NN for Machine Learning

Distance in the Feature Space

- Let $\phi(.)$ be the pulling function from the input space to the feature space (reproducing kernel Hilbert space). Recall the kernel trick:

$$\boldsymbol{x}^\top \boldsymbol{x} \mapsto \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}). \tag{2}$$

- The squared Euclidean distance between points in the feature space (reproducing kernel Hilbert space) is [1]:

$$\|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|_k^2 = k(\boldsymbol{x}_i, \boldsymbol{x}_i) + k(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2k(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{3}$$

- Proof:

$$\begin{aligned}
\|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|_k^2 &= \big(\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\big)^\top \big(\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\big) \\
&= \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_i) + \phi(\boldsymbol{x}_j)^\top \phi(\boldsymbol{x}_j) - \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j) - \phi(\boldsymbol{x}_j)^\top \phi(\boldsymbol{x}_i) \\
&\overset{(a)}{=} \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_i) + \phi(\boldsymbol{x}_j)^\top \phi(\boldsymbol{x}_j) - 2\phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j) \\
&\overset{(b)}{=} k(\boldsymbol{x}_i, \boldsymbol{x}_i) + k(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2k(\boldsymbol{x}_i, \boldsymbol{x}_j),
\end{aligned}$$

where (a) is because we can change the order of terms in inner product and (b) is because of the kernel trick.

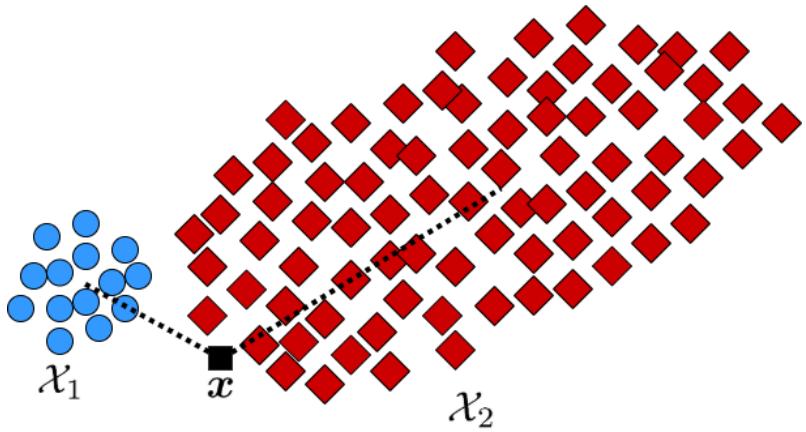
Kernel k-NN for Machine Learning

- For calculation of the distance of points, we can pull the points to the feature space and then calculate their distances in that feature space. Then, k-NN can be calculated using the distances in the feature space. This algorithm is called **kernel k-NN**.
- Kernel k-NN can be used for machine learning tasks similar to how we use k-NN for machine learning.

Distance Metric Learning for Large Margin Nearest Neighbor Classification

Generalized Mahalanobis Distance

- k-Nearest Neighbor (k-NN) classification is highly impacted by the distance metric utilized for measuring the differences between data points.
- Euclidean distance does not weight the points and it values them equally.



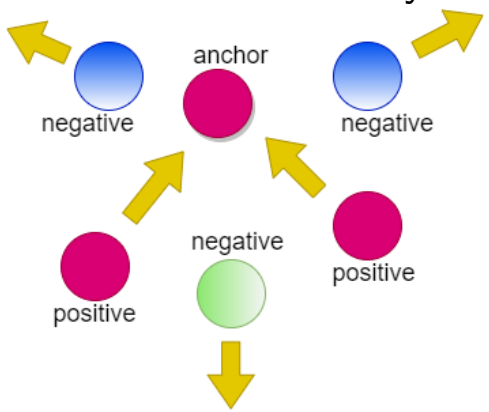
- A general distance metric can be viewed as the Euclidean distance after projection of points onto a discriminative subspace [2]. This projection can be viewed as a linear transformation with a projection matrix denoted by $\boldsymbol{L}$. We call this general metric the **generalized Mahalanobis distance** [2, 3, 4]:

$$\begin{aligned}\mathcal{D} &:= \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2_{\boldsymbol{M}} := \|\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)\|_2^2 = (\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j))^\top(\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)) \\ &= (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \boldsymbol{L}\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \boldsymbol{M}(\boldsymbol{x}_i - \boldsymbol{x}_j),\end{aligned} \tag{4}$$

where $\boldsymbol{M} := \boldsymbol{L}\boldsymbol{L}^\top$. The matrix $\boldsymbol{M}$ must be positive semi-definite, i.e. $\boldsymbol{M} \succeq 0$, for the metric to satisfy convexity and the triangle inequality [5].

Large Margin Nearest Neighbor Classification

- In order to improve the k-NN classification performance, we should decrease and increase the intra- and inter-class variances of data, respectively [6]. As can be seen in this figure, one way to achieve this goal is to pull the data points of the same class toward one another while pushing the points of different classes away.



- Let y_{il} be one (zero) if the data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_l$ are (are not) from the same class. Moreover, let η_{ij} be one if $\boldsymbol{x}_j$ is amongst the k-nearest neighbors of $\boldsymbol{x}_i$ with the same class label; otherwise, it is zero.
- For tackling the goal of pushing together the points of a class and pulling different classes away, the following cost function can be minimized [7]:

$$\sum_{i,j} \eta_{ij} \|\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)\|_2^2 + c \sum_{i,j,l} \eta_{ij}(1 - y_{il}) \Big[1 + \|\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)\|_2^2 - \|\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_l)\|_2^2\Big]_+, \tag{5}$$

where $[.]_+ := \max(., 0)$ is the **standard Hinge loss**. The first term in Eq. (5) pushes the same-class points towards each other. The second term, on the other hand, is a **triplet loss** [8] which increases and decreases the inter- and intra-class variances, respectively.

Large Margin Nearest Neighbor Classification

- Eq. (5) was:

$$\sum_{i,j} \eta_{ij} \|\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)\|_2^2 + c \sum_{i,j,l} \eta_{ij}(1 - y_{il}) \Big[1 + \|\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)\|_2^2 - \|\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_l)\|_2^2\Big]_+,$$

- Inspired by support vector machines, the cost function (5) can be restated using slack variables:

$$\begin{aligned} \underset{\boldsymbol{M},\, \xi_{ijl}}{\text{minimize}} \quad & \mathcal{L} := \sum_{i,j} \eta_{ij} \, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{M}}^2 + c \sum_{i,j} \eta_{ij} \, (1 - y_{il}) \, \xi_{ijl}, \quad \forall l \\ \text{subject to} \quad & \|\boldsymbol{x}_i - \boldsymbol{x}_l\|_{\boldsymbol{M}}^2 - \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{M}}^2 \geq 1 - \xi_{ijl}, \\ & \xi_{ijl} \geq 0, \\ & \boldsymbol{M} \succeq \boldsymbol{0}, \end{aligned} \tag{6}$$

  which is a SDP problem [9].

- The first term in the objective functions of Eqs. (5) and (6) are equivalent because of Eq. (4). The Hinge loss in Eq. (5) can be approximated using non-negative slack variables, denoted by ξ_{ijl}. The second term of objective function in Eq. (6), in addition to the first and second constraints, play the role of Hinge loss.

Large Margin Nearest Neighbor Classification

- We should solve this SDP problem using optimization toolboxes which use optimization algorithms such as the interior-point method.
- It is solved iteratively and usually slow to be solved. So, it is not used much in practice although its theory is solid.
- This method is called **large margin metric learning for nearest neighbor classification** [7, 10].
- I and my coauthors have a paper proposing **triplet mining** for this algorithm. See our paper "Acceleration of large margin metric learning for nearest neighbor classification using triplet mining and stratified sampling" [11].

References

[1] B. Schölkopf, "The kernel trick for distances," *Advances in neural information processing systems*, pp. 301–307, 2001.

[2] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Spectral, probabilistic, and deep metric learning: Tutorial and survey," *arXiv preprint arXiv:2201.09267*, 2022.

[3] B. Kulis *et al.*, "Metric learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2013.

[4] A. Globerson and S. T. Roweis, "Metric learning by collapsing classes," in *Advances in Neural Information Processing Systems*, pp. 451–458, 2006.

[5] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[6] B. Ghojogh, M. Sikaroudi, S. Shafiei, H. R. Tizhoosh, F. Karray, and M. Crowley, "Fisher discriminant triplet and contrastive losses for training siamese networks," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2020.

[7] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems*, pp. 1473–1480, 2006.

References (cont.)

[8] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[9] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.

[10] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[11] P. A. Poorheravi, B. Ghojogh, V. Gaudet, F. Karray, and M. Crowley, "Acceleration of large margin metric learning for nearest neighbor classification using triplet mining and stratified sampling," *arXiv preprint arXiv:2009.14244*, 2020.