CS419

Lecture 13: Similarity Functions & SVM Continued

3^{rd} March 2023

Lecturer: Abir De *Scribe: Priyansh, Aadi, Shantanu, Vishruth*

1 Similarity Functions

Similarity learning is an area of supervised machine learning in artificial intelligence. It is closely related to regression and classification, but the goal is to learn a similarity function that measures two objects' similarity. It has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification.

The computation of these similarity functions becomes too expensive when we have an input feature with large dimensions (like 1 million). To address this issue, we introduce new approximation problems. Once we solve this problem, the computation becomes much faster and less resource-intensive in terms of storage and time.

These functions & methods are used extensively by search engines (like Google) to produce search results from a large basket of webpages.

1.1 Dot Product Similarity

$$\boldsymbol{Sim(X_i, X_j) = X_i^T X_j}$$

The approximation problem is to find a function *h*:

$$\boldsymbol{Sim}(X_i, X_j) = X_i^T.X_j \simeq h(X_i^T).h(X_j)$$
$$\ni dim(h(X_k)) << dim(X_k)$$

More formally, this can be restated as:

$$\mathop{\mathbb{E}}_{h \in H}[h(X_i^T).h(X_j)] = X_i^T.X_j$$

1.2 Cosine Similarity

$$\boldsymbol{Sim(X_i, X_j) = \frac{X_i^T.X_j}{||X_i||.||X_j||}}$$

The approximation problem is to find X_1 & Y_1:

$$\frac{X^T.Y}{||X||.||Y||} \simeq \frac{X_1^T.Y_1}{||X_1||.||Y_1||}$$
$$\ni dim(X_1)) << dim(X) \quad \& \quad dim(Y_1)) << dim(Y)$$

More formally, this can be restated as:

$$\mathop{\mathbb{E}}_{X_1 \in D(x), Y_1 \in D(y)} \left[\frac{X_1^T.Y_1}{||X_1||.||Y_1||} \right] = \frac{X^T.Y}{||X||.||Y||}$$

One such example of X_1 & Y_1 is:

$$X_1 = \begin{bmatrix} \sqrt{1 - ||X||^2} \\ X \\ 0 \end{bmatrix}, \quad Y_1 = \begin{bmatrix} 0 \\ Y \\ \sqrt{1 - ||Y||^2} \end{bmatrix}$$

But this approximation does not solve the problem of computation time and storage. We can take X_1 as the following and Y_1 in a similar manner:

$$X_1 = W.X$$

where X_1, W & X are matrices of dimension $d \times 1$, $d \times n$, $n \times 1$ respectively $\ni d \ll n$
& values of W are sampled from the Normal Distribution

Upon substitution of X_1 & Y_1 in the approximation problem, we get:

$$\frac{X_1^T.Y_1}{||X_1||.||Y_1||} = \frac{X^T.W^T.W.Y}{||X||.||Y||} \approx \frac{X^T.Y}{||X||.||Y||}$$

as $W^T.W \approx I$ *(Identity Matrix)*

Note: Formulating efficient approximations for Cosine Similarity is easier than Dot Product Similarity but is more time intensive (primarily because of the calculations of norms)

1.3 Distance Similarity

Distance similarity is a measure of similarity between two objects based on their distance in some metric space. The distance function defines the distance between two objects in the metric space, and the similarity function is typically defined as a decreasing function of the distance. For example,

$$\boldsymbol{Sim(X_i, X_j) = k.e^{\frac{||X_i - X_j||^2}{\Delta}}}$$

Many distance functions can be used to compute distance similarity; the choice of distance function can significantly impact the results of a clustering algorithm. Different distance functions may be more appropriate for different data types or applications.

1.4 Generalising Similarity

Generalising similarity functions aims to capture the similarity between two objects by mapping them into a higher dimensional space, where the similarity can be more easily defined. This is done by applying a function ϕ to each object that maps it into a feature space. The similarity between two objects x_i and x_j can then be defined as the dot product of their feature space mappings, $\phi(x_i)^\top \phi(x_j)$. However, computing this dot product in the feature space can be computationally expensive or even impossible if the dimension of the feature space is too high.

To address this issue, generalising similarity functions introduce a new function ψ, which maps each object into a new space that is typically of lower dimensionality than the feature space. This allows us to compute the similarity between two objects using the dot product of their lower-dimensional mappings, $\psi(x_i)^\top \psi(x_j)$. The function ψ is typically chosen to have certain desirable properties, such as being finite-dimensional, and it should be chosen such that it preserves the similarity between objects in the original feature space. In other words, if two objects are similar in the original space, their mappings should also be similar in the new space.

One way to ensure that the function ψ preserves similarity is to choose it such that it is a valid kernel function. A kernel function is a function that satisfies the properties of positive definiteness and symmetry and can therefore be used to define a valid similarity measure between objects. In particular, if ψ is a valid kernel function, then the dot product of any pair of mappings $\psi(x_i)^\top \psi(x_j)$ can be interpreted as the similarity between the corresponding objects x_i and x_j. Therefore,

$$Sim(x_i, x_j) = \phi(x_i^T).\phi(x_j) \approx \psi(x_i^T).\psi(x_j)$$
$$\ni Sim(x_i, x_i) \geq 0 \quad \& \quad Sim(x_i, x_j) = Sim(x_j, x_i) \quad \& \quad dim(\psi) << dim(\phi)$$

More formally, this can be restated as:

$$\mathop{\mathbb{E}}_{\psi \in D(\psi)} [\psi(x_i^T).\psi(x_j)] = \phi(x_i^T).\phi(x_j)$$

1.5 Random Fourier Kernel

For a general similarity function, if we represent it as $\boldsymbol{Sim}(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ for some characteristic vector ϕ, then it could be an infinite vector. Therefore we want to approximate this function as $\psi(x_i)^T \psi(x_j)$ for some finite vector ψ. The way to do it is to use the inverse Fourier transform. We take the special case $\boldsymbol{Sim}(x_i, x_j) = F(x_i - x_j)$. Then,

$$F(x_i - x_j) = \int_{-\infty}^{\infty} g(w) e^{-\gamma w (x_i - x_j)} dw$$

where $\gamma^2 = -1$. We multiply and divide by some probability distribution p to get

$$\begin{aligned} F(x_i - x_j) &= \int_{-\infty}^{\infty} \frac{g(w)}{p(w)} e^{-\gamma w(x_i - x_j)} p(w) dw \\ &= \int_{-\infty}^{\infty} \left(\sqrt{\frac{g(w)}{p(w)}} e^{-\gamma w x_i} \right) \cdot \left(\sqrt{\frac{g(w)}{p(w)}} e^{\gamma w x_j} \right) p(w) dw \\ &= \mathop{\mathbb{E}}_{w \sim p(\cdot)} \left(\sqrt{\frac{g(w)}{p(w)}} e^{-\gamma w x_i} \right) \cdot \left(\sqrt{\frac{g(w)}{p(w)}} e^{\gamma w x_j} \right) \end{aligned}$$

Thus we have reduced F to an expectation and can use the Monte Carlo approximation to get:

$$F(x_i - x_j) \approx \frac{1}{|D|} \sum_{k \in D} \left(\sqrt{\frac{g(w_k)}{p(w_k)}} e^{-\gamma w_k x_i} \right) \cdot \left(\sqrt{\frac{g(w_k)}{p(w_k)}} e^{\gamma w_k x_j} \right)$$

for some large dataset D. Thu,s it is approximated by $\psi(x_i)^T \psi(x_j)$ for the finite vector

$$\psi(x) = \left[\sqrt{\frac{g(w_k)}{p(w_k)}} e^{\gamma w_k x_i} \right]_{k=1}^{|D|}$$

2 SVM Continued

A general SVM may have a loss function of the form:

$$\min_w \ f(y_1 w^T x_1, y_2 w^T x_2, \dots) + \lambda ||w||^2$$

We can write $x = u + v$ where u is perpendicular to span of $\{x_i\}$, and v is in span of $\{x_i\}$. In that case, $w^T x_i = v^T x_i$ because $u^T x_i = 0$, and $||w||^2 = ||u||^2 + ||v||^2 \geq ||v||^2$. Thus, replacing w with v will reduce the loss function. Therefore, the optimal w has the form $w* = \sum_{i \in D} \alpha_i y_i x_i$, even in this general case.

In case of a feature function $\phi(x)$, we have $w* = \sum_{i \in D} \alpha_i y_i \phi(x_i)$. The dimension of ϕ might be infinite. However, for classifying a new point, we need to find *sign* of $w^T \phi(x)$ (assume no bias).

$$\implies y = \textit{Sign} \left(\sum_{i \in D} \alpha_i y_i \phi^T(x_i) \phi(x) \right)$$

Here $\phi^T(x_i)\phi(x)$ has dimension 1, so we need to approximate this as $\psi^T(x_i)\psi(x)$ for some finite vector ψ. As discussed above, we can generally do this for a distance-based similarity function.