KMeans Kernel Classifier

Course: Math Behind ML

Karthik Kurugodu
B.Tech Mathematics and Computing
Indian Institute of Technology Hyderabad
ma20btech11008@iith.ac.in

Nikhil Kongara
B.Tech Mathematics and Computing
Indian Institute of Technology Hyderabad
ma20btech11011@iith.ac.in

***Abstract*—The least squares SVM is a kernel method for non-linear regression and classification tasks. Here we combine KMeans clustering with the least squares SVM. First KMeans clustering is used to extract a set of representative vectors for each class, and then LS-SVM uses these representative vectors as a training dataset for the classification task**

I. Introduction

The kernel methods transform a given non-linear problem into a linear one by using a similarity kernel function $\Omega(x, x\prime)$. It is a similarity function defined over pairs of input data points $(x, x\prime)$. This way the input data is mapped into a higher dimensional feature space $\phi(x)$, where the inner product $\langle \cdot\ ,\ \cdot \rangle$ can be calculated using Mercer's condition:

$$\Omega(x, x\prime) = \langle x\ ,\ x\prime \rangle \tag{1}$$

Consider $\chi = \{x_n | n = 1, \ldots, N\}$ as training dataset.
Representer theorem: Any non-linear function $f : \chi \longrightarrow \mathbb{R}$ can be expressed as linear combination of kernel products on training dataset which was mentioned above earlier.

$$f(x) = \sum_{n=1}^{N} a_n \Omega(x, x_n) \tag{2}$$

Time complexity of LS-SVM is $O(N^3)$ where N is size of the training dataset which is too high and makes it unsuitable for large dataset. So for this reason we use KMeans clustering to extract a set of representative vectors for each class, and then LS-SVM uses these representative vectors as a training dataset for the classification task. This way we can reduce the time complexity of LS-SVM to $O((KQ)^3)$ where K is the number of classes and Q is number of centroids in each class. These representative vectors are also called as **centroids**. These are then used by LS-SVM to classify the test data. This KMeans-LS-SVM method has some advantages:

1) It is faster than LS-SVM.
2) It is more robust.
3) It is very easy to implement.

II. Kernel LS-SVM Classifer

We already know that in binary classification, kernel SVM method constructs a hyperplane with the maximal margin between the two classes in feature space $\phi(x)$. This can be represented as convex quadratic programming problem involving inequality constraints.

The kernel LS-SVM simplifies the optimization problem by considering equality constraints only, such that solution is obtained by solving a system of linear equations. Now this problem is similar to ridge regression problem which is formulated as follows:

$$\min_{w,b} \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{n=1}^{N} (\hat{y}_n - w^T \phi(x_n) - b)^2 \tag{3}$$

Assume that K classes are encoded using standard basis in $\mathbb{R}^K$, i.e, let $x_i \in C_k$, then output y_i is a vector with 1 in the

k^{th} position and 0 elsewhere:

$$y_{ij} = \begin{cases} 1 & \text{if } x_i \in C_j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Consider input data $\{(x_i, yi) | x_i \in \mathbb{R}^{\mathbb{M}}, y_i \in \mathbb{R}^{\mathbb{K}}, i = 1, \ldots, N\}$ and the feature mapping function $\phi(x)$. The kernel LS-SVM is formulated as follows:

$$\min_{w,b} S(w, b, \epsilon) = \frac{1}{2} \sum_{j=1}^{K} w_j^T w_j + \frac{\gamma}{2} \sum_{i=1}^{N} \sum_{j=1}^{K} \epsilon_{ij}^2 \tag{5}$$

subject to

$$\langle \phi(x)\ ,\ \omega_j \rangle + b_j = y_{ij} - \epsilon_{ij}, i = 1, \ldots, N; j = 1, \ldots, K \tag{6}$$

$$w_j^T \phi(x_i) + b_j = y_{ij} - \epsilon_{ij}, i = 1, \ldots, N; j = 1, \ldots, K \tag{7}$$

where $\epsilon_{ij} \geq 0$ are approximation errors, b_j is bias coefficient, $w^{(j)}$ is the vector of weights corresponding to the j^{th} class. The objective function S is a sum of least squares errors and the regularization term. This regularization parameter γ corresponds to a multi-dimensional version of the ridge regression problem.

In the primal weight space the multi class classifier takes the form:

$$x \in C_k, \Leftrightarrow k = arg \max_{j=1,\ldots,K} g_j(x)$$

$$\text{where } g_j(x) = \frac{\exp(\langle \phi(x)\ ,\ w^{(j)} \rangle) + b_j)}{\sum_{i=1}^{K} \exp(\langle \phi(x)\ ,\ w^{(i)} \rangle) + b_i)}$$

Here g_j is the non-linear soft max function

Now applying Lagrangian to (5)

$$\begin{aligned} L(w, b, \epsilon, a) &= S(w, b, \epsilon) \\ &- \sum_{i=1}^{N} \sum_{j=1}^{K} a_{ij} [\langle \phi(x)\ ,\ \omega_j \rangle + b_j - y_{ij} + \epsilon_{ij}] \end{aligned}$$

where $a_{ij} \in \mathbb{R}$ is the Lagrange multiplier. Now applying KKT conditions:

$$\frac{\partial L}{\partial w^{(j)}} = 0 \implies w^{(j)} = \sum_{n=1}^{N} a_{nj} \phi(x_n) \tag{8}$$

$$\frac{\partial L}{\partial b_{(j)}} = 0 \implies \sum_{i=1}^{N} a_{ij} = 0 \tag{9}$$

$$\frac{\partial L}{\partial \epsilon_{ij}} = 0 \implies a_{ij} = \gamma \epsilon_{ij} \tag{10}$$

$$\frac{\partial L}{\partial a_{ij}} = 0 \implies \langle \phi(x)\ ,\ \omega_j \rangle + b_j - y_{ij} + \epsilon_{ij} = 0 \tag{11}$$

Now from eq(8), eq(10) and eq(11):

$$\sum_{n=1}^{N} [\Omega(x_i, x_n) + \gamma^{-1} \delta_{in}] a_{nj} + bj = y_{ij}, \tag{12}$$

Here δ_{in} is the Kronecker delta function: where $\delta_{in} = 1$ if $i = n$ and 0 otherwise

As you can see in eq(12) there are K independent system of equations with binary labels y_{ij}. Now each system can be written in the matrix form as follows:

$$\begin{bmatrix} 0 & u^T \\ u & \Omega + \gamma^{-1} I \end{bmatrix} \begin{bmatrix} b_j \\ a^{(j)} \end{bmatrix} = \begin{bmatrix} 0 \\ y_j \end{bmatrix}, j = 1, \ldots, K \tag{13}$$

Here $I_{N \times N}$ is the identity matrix, $u_{N \times 1} = [1, \ldots, 1]^T$ is a vector of ones, $a_{N \times 1}^{(j)} = [a_{1j}, \ldots, a_{Nj}]^T$ is weights and $y_j = [y_{1j}, \ldots, y_{Nj}]^T$ is the vector of binary labels for the j^{th} class. Each system has $N+1$ linear equations with $N+1$ unknowns.

$$\Theta = \begin{bmatrix} 0 & u^T \\ u & \Omega + \gamma^{-1} I \end{bmatrix} \tag{14}$$

All the K systems can be written as:

$$\Theta W = Z \tag{15}$$

where

$$W_{(N+1) \times K} = \begin{bmatrix} b_1 & \ldots & b_K \\ a^{(1)} & \ldots & a^{(K)} \end{bmatrix}, Z_{(N+1) \times K} = \begin{bmatrix} 0 & \ldots & 0 \\ y_1 & \ldots & y_K \end{bmatrix}$$

Now once all the K systems are solved, we consider multi-class classifier in dual space(from eq (12)) as follows:

$$g_j(x) = \frac{\exp(\langle \phi(x)\ ,\ w^{(j)} \rangle) + b_j)}{\sum_{i=1}^{K} \exp(\langle \phi(x)\ ,\ w^{(i)} \rangle) + b_i)}$$

Substituting eq(8) in above equation

$$g_j(x) = \frac{\prod_{n=1}^{N} \exp(\Omega(x, x_n) a_{nj} + b_j)}{\sum_{i=1}^{K} \prod_{n=1}^{N} \exp(\Omega(x, x_n) a_{ni} + b_i)}$$

Now our problem becomes:

$$x \in C_k, \Leftrightarrow k = arg \max_{j=1,\ldots,K} g_j(x)$$

$$\text{where } g_j(x) = \frac{\prod_{n=1}^{N} \exp(\Omega(x, x_n)a_{nj} + b_j)}{\sum_{i=1}^{K} \prod_{n=1}^{N} \exp(\Omega(x, x_n)a_{ni} + b_i)}$$

Here g_j is the non-linear soft max function

III. KMeans Clustering

First we use KMeans clustering algorithm to extract a set of representative vectors for each class. Now this representative vectors will be passed into LS-SVM kernel model as training dataset. KMeans clustering algorithm is as follows:

1) Take $\{x_i^k | x_i^k \in \mathbb{R}^{\mathbb{M}}, i = 1, \ldots, N_k\}$ as training samples for class C_k where N_k is the number of training samples for the class C_k and $N = \sum_{k=1}^{K} N_k$ is the total number of training samples.
2) Take $\{\mu_q^k | \mu_q^k \in \mathbb{R}^{\mathbb{M}}, q = 1, \ldots, Q\}$ as initial centroids for class C_k where $Q < N_K$ is the number of centroids for class C_k.
3) Build a matrix $X_k = [x_{im}^k]_{N_k \times M}$ where each row is a training sample for class C_k.
4) Build a matrix $M_k = [\mu_{qm}^k]_{Q \times M}$ where each row is a randomly initialized centroid for class C_k.
5) Let $R_k = X_k M_k^T = [r_{iq}^k]_{N_k \times Q}$
6) Let $\hat{R_k} = [\hat{r_{iq}^k}]_{N_k \times Q}$ be transformed sparse matrix of R_k where:

$$\hat{r_{iq}^k} = \begin{cases} 1 & \text{if } q = \arg\max_q r_{iq}^k \\ 0 & \text{otherwise} \end{cases} \quad i = 1, \ldots, N_k$$

Each sample is assigned to the nearest centroid.
7) $\hat{M_k} = \hat{R_k^T} X_k = [\hat{\mu_{qm}^k}]_{Q \times M}$.
This is the new set of centroids.
8) Normalizing new set of centroids:

$$\hat{\mu_q^k} = \frac{\hat{\mu_q^k}}{||\hat{\mu_q^k}||} \qquad q = 1, \ldots, Q$$

9) Computing alignment deviation between new set and old set of centroids:

$$\delta = 1 - \frac{\sum_{q=1}^{Q} \langle \hat{\mu_q^k} \;\; \mu_q^k \rangle}{Q}$$

10) $M_k = \hat{M_k}$
11) Repeat steps 5 to 10 until $\delta < \beta$ where β is the tolerance.
12) Return M_k

Here β is considered as small as possible.

IV. KMeans Kernel LS-SVM Classifier

After extracting a set of representative vectors for each class $C_k, k = 1, \ldots, K$ using KMeans clustering, we pass these KQ centroids into LS-SVM kernel model as training dataset. Training dataset for LS-SVM before KMeans clustering:

$$\{(x_i^k, y_i^k) | x_i^k \in \mathbb{R}^{\mathbb{M}}, y_i^k \in \mathbb{R}^{\mathbb{K}}, i = 1, \ldots, N\}$$

Training dataset for LS-SVM after KMeans clustering:

$$\{(\mu_q^k, y_q^k) | \mu_q^k \in \mathbb{R}^{\mathbb{M}}, y_q^k \in \mathbb{R}^{\mathbb{K}}, q = 1, \ldots, KQ\}$$

As you can see the training dataset size is reduced from N to KQ where $KQ < N$.

Previously there were $N + 1$ linear equations with $N + 1$ unknowns and $O(N^3)$ time complexity.

Now there are $KQ + 1$ linear equations with $KQ + 1$ unknowns and $O((KQ)^3)$ time complexity.
As we discussed earlier our problem previously was:

$$x \in C_k, \Leftrightarrow k = arg \max_{j=1,\ldots,K} g_j(x)$$

$$\text{where } g_j(x) = \frac{\prod_{n=1}^{N} \exp(\Omega(x, x_n)a_{nj} + b_j)}{\sum_{i=1}^{K} \prod_{n=1}^{N} \exp(\Omega(x, x_n)a_{ni} + b_i)}$$

Now our problem becomes:

$$x \in C_k, \Leftrightarrow k = arg \max_{j=1,\ldots,K} g_j(x)$$

$$\text{where } g_j(x) = \frac{\prod_{n=1}^{KQ} \exp(\Omega(x, \mu_n^k)a_{nj} + b_j)}{\sum_{i=1}^{K} \prod_{n=1}^{KQ} \exp(\Omega(x, \mu_n^k)a_{ni} + b_i)}$$

Here g_j is the non-linear soft max function

V. Application

We have implemented the KMeans Kernel LS-SVM Classifier on the MNIST dataset. It is a dataset of handwritten digits $(0, 1, \ldots, 9)$ consisting of 60000 training images and 10000 test images. Each image is of size 28×28 pixels with monochrome color with intensity ranging from 0 to 255. Here are the following steps we have followed:

1) Extracting all the overlapping patches of size $l \times l$ by iterating over all images x_n from a class C_k where $l < L = 28$.
2) Vectorize all the patches by concatenating columns and normalize.

$$x_{nij} = x_{nij} - avg(x_{nij})$$
$$x_{nij} = \frac{x_{nij}}{||x_{nij}||}$$

3) Now these patches are used to define the input data for KMeans clustering algorithm.
4) After KMeans clustering, we get KQ centroids for each class C_k.
5) Now these centroids are used to define the input data for LS-SVM kernel model. With each centroid we associate a label y_q^k which is a vector with 1 in the k^{th} position and 0 elsewhere and we train the LS-SVM classifier.

A. Explanation

From each image x_n we are extracting $(L-l+1) \times (L-l+1)$ patches. We're sliding this patch across the image one pixel at a time, both horizontally and vertically. Horizontally, we can start the patch at any of the first $(L - l + 1)$ columns.

For example, if L is 5 and l is 3, we can start the patch at 1^{st} 2^{nd} or 3^{rd} column. That's $5 - 3 + 1 = 3$ possible starting columns. The same logic applies vertically, giving us $(L-l+1)$ possible starting rows. Since we can independently choose the starting row and the starting column, we multiply these two quantities together to get the total number of patches. Hence, the formula for the number of patches is $(L-l+1) \times (L-l+1)$.

B. Training

In the code we have taken $l = 25$. Hence, we get $(28 - 25 + 1) \times (28 - 25 + 1) = 4 \times 4 = 16$ patches from each image. Therefore we get $16 \times 60,000 = 9,60,000$ training samples(patches) for LS-SVM classifier. Considered $100 \leq Q \leq 5000$, $K = 10$, $\gamma = 10^{-6}$ i.e., regularization parameter and $\beta = 10^{-6}$ i.e., tolerance for KMeans clustering.

For this particular dataset, polynomial kernel is considered as degree of 4:

$$\Omega(x, x\prime) = \langle x \ , \ x\prime \rangle^4$$

C. Testing

After training the LS-SVM classifier, we test the classifier on the test dataset. Let x_n be a test image. We extract all the overlapping patches of size $l \times l$ from the test image x_n. Now we pass these patches into the LS-SVM classifier and get the predicted label for each patch. We take the majority vote of all the predicted labels, assign that label to the test image x_n.

D. Interpretation

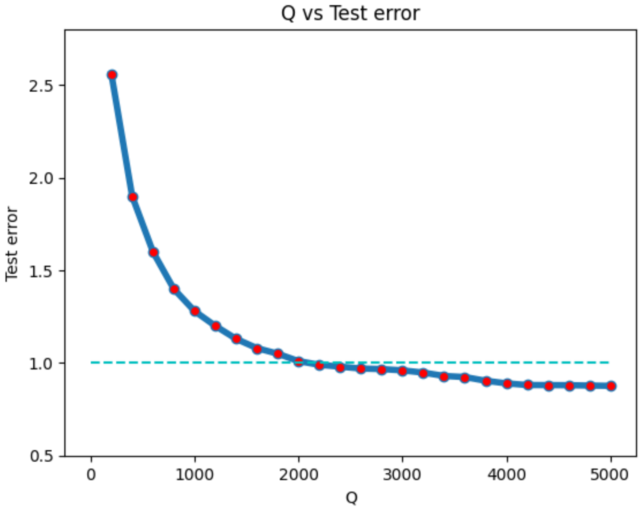

Fig. 1: Q vs $\eta\%$ (test error)

From the Fig. **??** optimal Q was considered as 4000 where the test error, $\eta = 0.89\%$.

VI. Conclusion

First we have used KMeans clustering algorithm to extract a set of representative vectors for each class. Then we have passed these representative vectors into LS-SVM kernel model as training dataset. This way we have reduced the time complexity of LS-SVM from $O(N^3)$ to $O((KQ)^3)$ where K is the number of classes and Q is number of centroids in each class. This method is significantly faster than LS-SVM, it is more robust and easy to implement.