Large Margin Nearest Neighbor Bounds

Ryan Curtin

June 15, 2018

When we run LMNN, we are learning a distance metric L. Here I will use the notation L where $L^T L = M$ as in the original paper. Given some dataset X made up of points $\{x_0, x_1, \ldots, x_n\}$ with labels $y = \{y_0, y_1, \ldots, y_n\}$. We also have a parameter k specified by the user (this is the number of neighbors).

At each step of the optimization, we must calculate the k nearest neighbors of each point x_i. We need k points of the same label (targets), and k points of another label (impostors).

However, if L does not change significantly, then the k nearest impostors are not likely to change. Therefore, we wish to find some bounds that will show us when we do not need to recalculate the impostors.

Let us consider that we have the following quantities:

- x_i: the query point
- x_a: the k-th nearest impostor to x_i
- x_b: any other impostor not in the top k for x_i
- L_t: the L transformation matrix at iteration t
- L_{t+1}: the L transformation matrix at iteration $t + 1$

We also know the following quantities:

- $d_{L_t}(x_i, x_a)$
- $d_{L_t}(x_i, x_b)$
- $d_{L_{t+1}}(x_i, x_a)$
- $d_{L_{t+1}}(x_i, x_b)$

Let's assume that

$$d_{L_{t+1}}(x_i, x_a) < d_{L_{t+1}}(x_i, x_b). \tag{1}$$

This would mean that we do not need to recalculate impostors the following iteration. Starting with this assumption, we can rework the expression by plugging L into the arguments of the distance function:

$$d_{L_{t+1}}(x_i, x_b) = d(L_{t+1}x_i, L_{t+1}x_b). \tag{2}$$

Then, we can use the reverse triangle inequality to bound the distance from below. Note that the two terms on the right hand side of the reverse triangle inequality could actually be reversed. We'll come back to that later.

$$\begin{aligned} d(L_{t+1}x_i, L_{t+1}x_b) &\geq d(L_t x_i, L_{t+1}x_b) - d(L_t x_i, L_{t+1}x_i) && (3) \\ &\geq d(L_t x_i, L_t x_b) - d(L_t x_b, L_{t+1}x_b) - d(L_t x_i, L_{t+1}x_i). && (4) \end{aligned}$$

Similarly, we can bound the distance from above for the other side of the conditional with the triangle inequality.

$$
\begin{aligned}
d(L_{t+1}x_i, L_{t+1}x_a) &\leq d(L_t x_i, L_{t+1}x_a) + d(L_t x_i, L_{t+1}x_i) && (5)\\
&\leq d(L_t x_i, L_t x_a) + d(L_t x_a, L_{t+1}x_a) + d(L_t x_i, L_{t+1}x_i). && (6)
\end{aligned}
$$

It would be expensive to calculate the exact distance $d(L_t x_i, L_{t+1}x_i)$, but we could express this differently.

$$
\begin{aligned}
d(L_t x_i, L_{t+1}x_i) &= (L_t x_i - L_{t+1}x_i)^T(L_t x_i - L_{t+1}x_i) && (7)\\
&= x_i^T(L_t - L_{t+1})^T(L_t - L_{t+1})x_i && (8)\\
&= x_i^T\|L_t - L_{t+1}\|_F^2 x_i && (9)\\
&= \|L_t - L_{t+1}\|_F^2\|x_i\|^2. && (10)
\end{aligned}
$$

Now we may plug this back in to our original conditional along with the other bounds:

$$
\begin{aligned}
d_{L_{t+1}}(x_i, x_a) &< d_{L_{t+1}}(x_i, x_b) && (11)\\
d_{L_t}(x_i, x_a) + \|L_t - L_{t+1}\|_F^2(\|x_i\|^2 + \|x_a\|^2) &< d_{L_t}(x_i, x_b) - \|L_t - L_{t+1}\|_F^2(\|x_i\|^2 + \|x_b\|^2) && (12)\\
\|L_t - L_{t+1}\|_F^2(2\|x_i\|^2 + \|x_a\|^2 + \|x_b\|^2) &< d_{L_t}(x_i, x_b) - d_{L_t}(x_i, x_a). && (13)
\end{aligned}
$$

We can use this final result. If we cache $d_{L_t}(x_i, x_a)$ and $d_{L_t}(x_i, x_b)$, as well as $\|x_i\|^2 \forall x_i$, we need only calculate $\|L_t - L_{t+1}\|_F^2$ at each iteration, and we can then quickly check to see if the condition is violated for each query point p_i. There are a few options for how it could be used:

- Check each query point individually, in order to assemble a smaller query set for the nearest neighbor search.
- Check all points and just run the search if the condition is violated for any point p_i.

Which of those strategies is better to use will depend on how often the bounds are typically violated with different optimizers. Also, note that the bound itself is dependent on the squared norm of the points $\|x_i\|^2$. This means that it would be wise to mean-center the points, which will reduce the norms $\|x_i\|^2$ but not change the nearest neighbor search results. (In fact, if the authors are suggesting to perform PCA before the LMNN process, typically we will get centered points as a result anyway.)