Margin-Enforcing Projection

Su Jianlin

June 19, 2026

This article introduces a mathematical operation named "Margin-Enforcing Projection (MEP)", which divides a vector into two parts in a specified manner, and then requires the margin between these two parts to be at least m (Margin).

1 Background

We know that classification tasks aim to select the correct class, so the training objective is usually simply that the positive-class score is greater than the negative-class score. However, in some scenarios, we not only hope that the positive-class score exceeds the negative-class score, but also hope that it exceeds by at least a specified margin $m > 0$.

These scenarios mainly fall into two types. The first type hopes for more robust classification results that are not easily disturbed by random noise, especially in low-precision inference scenarios; therefore, we hope that the margin of the prediction results can be more pronounced. The second type does not use a classification model at all, but instead intends to learn features through classification, with the final application being retrieval using features. If no margin is set at this time, the retrieval results near the boundary are very prone to errors.

In fact, these two scenarios were already very mature in earlier years, especially the second scenario, whose typical representative is the training of face-recognition feature models; therefore, this article can be regarded as "digging up an old post." The standard approach to this problem is to design various Margin Losses, such as Hinge Loss, Margin Softmax, AM-Softmax, etc. We briefly introduced them in *Sentence Similarity Model Based on GRU and AM-Softmax* and *From Triangle Inequality to Margin Softmax*.

The idea of this article is that if we have some operation that can project the predicted score $\boldsymbol{x}$ to a score $\boldsymbol{y}$ meeting the margin requirement, then we can directly let the model learn it as a target, for example by minimizing $\|\boldsymbol{y} - \boldsymbol{x}\|_2^2$. And this projection operation is the problem we will study next.

2 Mathematical Definition

Let $\boldsymbol{x} = (x_1, x_2, \cdots, x_n), \boldsymbol{z} = (z_1, z_2, \cdots, z_n)$, where $\boldsymbol{x}$ represents the model's predicted scores. For simplicity, we assume that the first k scores represent positive-class scores and the remaining $n - k$ represent negative-class scores. We define

$$\mathcal{P}_m(\boldsymbol{x}) \triangleq \operatorname*{argmin}_{\boldsymbol{z}\in\mathbb{R}^n} d(\boldsymbol{z}, \boldsymbol{x}) \quad \text{s.t.} \quad \min(\boldsymbol{z}_{\leq k}) - \max(\boldsymbol{z}_{>k}) \geq m \tag{1}$$

where $\boldsymbol{z}_{\leq k} = (z_1, \cdots, z_k), \boldsymbol{z}_{>k} = (z_{k+1}, \cdots, z_n)$, k is an integer greater than 0 and less than n, $m > 0$ is a given margin, and $d(\boldsymbol{z}, \boldsymbol{x})$ is the distance function to be minimized. We have two choices: L1 distance and L2 distance, which will be discussed separately below.

This definition is very intuitive: under the premise of satisfying the margin requirement, find the vector closest to the predicted score, which is consistent with the general idea of projection

operations. Therefore, we call it "Margin-Enforcing Projection (MEP)." Using such a projection result as a learning target can theoretically guide the model along the easiest and fastest path, and also allow it to "brake" in time after reaching the goal to avoid overtraining.

3 Common Results

If $\boldsymbol{x}$ already satisfies $\min(\boldsymbol{x}_{\leq k}) - \max(\boldsymbol{x}_{>k}) \geq m$, then obviously $\boldsymbol{z}^* = \boldsymbol{x}$, which is the trivial case. Without loss of generality, in what follows we assume $\min(\boldsymbol{x}_{\leq k}) - \max(\boldsymbol{x}_{>k}) < m$. It is not difficult to see that regardless of whether d chooses the L1 or L2 distance, the optimal solution $\boldsymbol{z}^*$ necessarily satisfies

$$\min(\boldsymbol{z}^*_{\leq k}) - \max(\boldsymbol{z}^*_{>k}) = m \tag{2}$$

otherwise one could always make some z_i closer to x_i to reduce the objective value. Based on this observation, let $\min(\boldsymbol{z}^*_{\leq k}) = \ell$, then $\max(\boldsymbol{z}^*_{>k}) = \ell - m$, and we can obtain

$$\boldsymbol{z}^*_{\leq k} = \max(\boldsymbol{x}_{\leq k}, \ell), \qquad \boldsymbol{z}^*_{>k} = \min(\boldsymbol{x}_{>k}, \ell - m) \tag{3}$$

At this point

$$\begin{aligned}\boldsymbol{z}^* - \boldsymbol{x} &= [\max(\boldsymbol{x}_{\leq k}, \ell) - \boldsymbol{x}_{\leq k}, \min(\boldsymbol{x}_{>k}, \ell - m) - \boldsymbol{x}_{>k}] \\ &= [\max(\ell - \boldsymbol{x}_{\leq k}, 0), \min(\ell - m - \boldsymbol{x}_{>k}, 0)] \\ &= [\max(\ell - \boldsymbol{x}_{\leq k}, 0), -\max(\boldsymbol{x}_{>k} + m - \ell, 0)]\end{aligned} \tag{4}$$

Next, we need to solve for ℓ according to the different choices of d.

4 L2 Distance

First consider the L2 distance. At this point we have

$$\|\boldsymbol{z}^* - \boldsymbol{x}\|_2^2 = \sum_{i=1}^{k} \max(\ell - x_i, 0)^2 + \sum_{j=k+1}^{n} \max(x_j + m - \ell, 0)^2 \triangleq f(\ell) \tag{5}$$

Taking the derivative yields

$$f'(\ell) = 2\sum_{i=1}^{k} \max(\ell - x_i, 0) - 2\sum_{j=k+1}^{n} \max(x_j + m - \ell, 0) \tag{6}$$

$$f''(\ell) = 2\#\{\ell > x_i\} + 2\#\{\ell < x_j + m\} \tag{7}$$

where $\#$ is the counting function, with the convention $1 \leq i \leq k < j \leq n$. Obviously $f''(\ell) \geq 0$, but we can further strengthen this to $f''(\ell) > 0$.

This is because $f''(\ell) = 0$ would imply $\#\{\ell > x_i\} = 0$ and $\#\{\ell < x_j + m\} = 0$, i.e., both $\min(\boldsymbol{x}_{\leq k}) \geq \ell$ and $\max(\boldsymbol{x}_{>k}) \leq \ell - m$ hold simultaneously, which contradicts the assumption $\min(\boldsymbol{x}_{\leq k}) - \max(\boldsymbol{x}_{>k}) < m$. Therefore $f''(\ell) > 0$, meaning $f(\ell)$ is strictly convex. Combined with the continuity of $f(\ell)$ and $f'(\ell)$, as well as $f'(-\infty) = -\infty$ and $f'(\infty) = \infty$, it follows that $f(\ell)$ has a unique minimizer, which must be attained at $f'(\ell) = 0$.

Since $f'(\ell)$ is a piecewise linear function composed of $\max(x, 0)$, $f'(\ell)$ is also a piecewise linear function of ℓ, whose breakpoints are all the points in $\{x_1, \cdots, x_k, x_{k+1} + m, \cdots, x_n + m\}$. To solve $f'(\ell) = 0$, we first sort these breakpoints $\{x_1, \cdots, x_k, x_{k+1} + m, \cdots, x_n + m\}$ in ascending order, obtaining $n - 1$ intervals. Within a single interval, $f'(\ell)$ is a straight line. Traversing all intervals $[a, b]$, we find the interval satisfying $f'(a) \leq 0$ and $f'(b) \geq 0$, and then compute the zero of this straight line within that interval.

5 L1 Distance

Next consider the L1 distance. At this point we have

$$\|\boldsymbol{z}^* - \boldsymbol{x}\|_1 = \sum_{i=1}^{k} \max(\ell - x_i, 0) + \sum_{j=k+1}^{n} \max(x_j + m - \ell, 0) \triangleq g(\ell) \tag{8}$$

Clearly, $g(\ell)$ is itself a piecewise linear function. The minimum of such a function can only be attained at the breakpoints, so the most straightforward solution is to enumerate all breakpoints $\{x_1, \cdots, x_k, x_{k+1}+m, \cdots, x_n+m\}$ and take the one that minimizes $g(\ell)$, with complexity $\mathcal{O}(n^2)$. But we can simplify further. First, take the derivative:

$$\begin{aligned} g'(\ell) &= \#\{\ell > x_i\} - \#\{\ell < x_j + m\} \\ &= \#\{\ell > x_i\} + \#\{\ell \geq x_j + m\} - (n - k) \end{aligned} \tag{9}$$

The second equality uses the identity $\#\{\ell < x_j + m\} + \#\{\ell \geq x_j + m\} = n - k$. From this it is easy to see that $g'(\ell)$ is monotonically increasing from negative to positive. Of course, $g'(\ell)$ is not continuous, so we cannot guarantee finding a point where $g'(\ell) = 0$.

However, if we change $\#\{\ell > x_i\}$ to $\#\{\ell \geq x_i\}$ (the derivative at a discontinuous breakpoint can be regarded as arbitrary, so this adjustment is permissible), then the meaning of $g'(\ell) = 0$ becomes exactly that "the number of breakpoints less than or equal to ℓ is exactly $n - k$." If we further assume that all breakpoints are distinct, then ℓ is exactly the $(n-k)^{\text{th}}$ smallest among all sorted breakpoints! Therefore, in the L1 scenario, a single sort is sufficient to obtain ℓ, which is rather elegant.

6 Reference Implementation

Reference implementations for both versions of MEP are as follows:

```
import jax
import jax.numpy as jnp

@jax.jit
def l2mep(inputs, mask, margin):
    x, m = inputs, margin
    u = jnp.where(mask, x, x + m).sort()[:, None]
    v = jnp.where(mask, jnp.fmax(u - x, 0), -jnp.fmax(x + m - u, 0)).sum(axis=1)
    i = ((v[:-1] < 0) & (v[1:] >= 0)).argmax()
    l = (u[i + 1] * v[i] - u[i] * v[i + 1]) / (v[i] - v[i + 1])
    return jnp.where(mask, jnp.fmax(x, l), jnp.fmin(x, l - m))

@jax.jit
def l1mep(inputs, mask, margin):
    x, m = inputs, margin
    u = jnp.where(mask, x, x + m).sort(axis=-1)
    l = jnp.take_along_axis(u, (~mask).sum(axis=-1, keepdims=True) - 1)
    return jnp.where(mask, jnp.fmax(x, l), jnp.fmin(x, l - m))
```

Here is a brief explanation. In actual scenarios, positive classes are usually not simply arranged in the first k positions; their number and position may be uncertain. Therefore, in the above implementation, we use a mask vector to mark positive and negative classes.

The current implementation of `l2mep` only supports 1D inputs. If batch dimensions need to be supported, simply wrap it with a layer of `jax.vmap`. This algorithm finds the zero-crossing interval by traversing all intervals. The per-step complexity is $\mathcal{O}(n)$, and the total complexity is

$\mathcal{O}(n^2)$. If efficiency needs to be improved, a binary search can be used to locate the zero-crossing interval, reducing the complexity to $\mathcal{O}(n \log n)$.

Since the algorithm of `l1mep` is simple, the current implementation already supports arbitrary batch dimensions. Its core operation is only sorting, with complexity $\mathcal{O}(n \log n)$. Whether judged by simplicity or speed, it is already quite ideal. If there are no other considerations, the L1 version is recommended in practice.

7 Summary

This article introduced a mathematical operation called "Margin-Enforcing Projection (MEP)": given a vector of scores, project it to the nearest vector satisfying "the minimum positive-class score is at least m greater than the maximum negative-class score." This can provide some new ideas for traditional Margin Learning.