Margin-Enforcing Projection

Su Jianlin

June 19, 2026

In this article, we introduce a mathematical operation called "Margin-Enforcing Projection (MEP)," which partitions a vector into two parts in a specified way and then requires the gap between these two parts to be at least m (the margin).

1 Background

We know that the goal of a classification task is to select the correct category, so the training objective is usually simply that "the positive-class score is greater than the negative-class score." However, in certain scenarios, we not only want the positive-class score to exceed the negative-class score, but also to exceed it by at least a specified margin $m > 0$.

These scenarios mainly fall into two types. The first is when we want the classification results to be more robust and not easily disturbed by random noise, especially in low-precision inference scenarios, so we prefer the margin of the prediction results to be more pronounced. The second is when we do not actually use a classification model at all, but rather use classification as a means to learn features, and the final scenario is to use those features for retrieval—in this case, if no margin is set, the retrieval results near the boundary are prone to errors.

In fact, both of these scenarios have been well established for many years, especially the second one, whose typical representative is the training of face-recognition feature models, so this article is in a sense a revisit of an old topic. The standard approach to this problem is to design various margin losses, such as Hinge Loss, Margin Softmax, AM-Softmax, etc., which we have briefly introduced in Sentence Similarity Model Based on GRU and AM-Softmax and From Triangle Inequality to Margin Softmax.

The idea of this article is: if we have some operation that can project the predicted scores $\boldsymbol{x}$ into scores $\boldsymbol{y}$ that satisfy the margin requirement, then we can simply use it as the target for the model to learn, for example by minimizing $\|\boldsymbol{y} - \boldsymbol{x}\|_2^2$. This projection operation is the problem we will study in what follows.

2 Mathematical Definition

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

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

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

This definition is very intuitive: among all vectors satisfying the margin requirement, find the one closest to the predicted scores. This conforms to the general idea of a projection

operation, so we call it "Margin-Enforcing Projection (MEP)." Using such a projection result as the learning target can, in theory, guide the model along the easiest and fastest path, while also allowing it to "brake" in time once the target is reached, avoiding overtraining.

3 Common Results

If $\boldsymbol{x}$ already satisfies $\min(\boldsymbol{x}_{\le k}) - \max(\boldsymbol{x}_{>k}) \ge m$, then clearly $\boldsymbol{z}^* = \boldsymbol{x}$, which is the trivial case. Without loss of generality, below we assume $\min(\boldsymbol{x}_{\le k}) - \max(\boldsymbol{x}_{>k}) < m$. It is not difficult to see that regardless of whether d is chosen as the L1 or the L2 distance, the optimal solution $\boldsymbol{z}^*$ must be attained at

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

otherwise one could always move some z_i closer to x_i to decrease the objective value. Based on this observation, let $\min(\boldsymbol{z}^*_{\le k}) = \ell$, so that $\max(\boldsymbol{z}^*_{>k}) = \ell - m$. Then we can obtain

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

and hence

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

The next step is to solve for ℓ according to the specific choice of d.

4 The L2 Distance

Let us first consider the L2 distance. In this case 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 derivatives, we obtain

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

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

where $\#$ is the counting function, with the convention $1 \le i \le k < j \le n$. Clearly $f''(\ell) \ge 0$, but this can be strengthened to $f''(\ell) > 0$.

This is because $f''(\ell) = 0$ would imply $\#\{\ell > x_i\} = 0$ and $\#\{\ell < x_j + m\} = 0$ simultaneously, i.e., $\min(\boldsymbol{x}_{\le k}) \ge \ell$ and $\max(\boldsymbol{x}_{>k}) \le \ell - m$, which contradicts the assumption $\min(\boldsymbol{x}_{\le k}) - \max(\boldsymbol{x}_{>k}) < m$. Therefore $f''(\ell) > 0$, meaning $f(\ell)$ is strictly convex. Combined with the continuity of $f(\ell)$ and $f'(\ell)$, along with $f'(-\infty) = -\infty$ and $f'(\infty) = \infty$, we can conclude that the minimizer of $f(\ell)$ is unique and must be attained where $f'(\ell) = 0$.

Since $f'(\ell)$ is a composition of the piecewise linear function $\max(x, 0)$, $f'(\ell)$ is itself a piecewise linear function of ℓ, whose breakpoints are all the x_i and $x_j + m$. To solve $f'(\ell) = 0$, we first sort the breakpoints $\{x_1, \cdots, x_k, x_{k+1} + m, \cdots, x_n + m\}$ in ascending order to obtain $n - 1$ intervals. Within each individual interval, $f'(\ell)$ is a straight line. We iterate over all intervals $[a, b]$ to find the one with $f'(a) \le 0$ and $f'(b) \ge 0$, and then compute the zero of the line within that interval.

5 The L1 Distance

Next, let us consider the L1 distance. In this case 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, $g(\ell)$ itself is a piecewise linear function, and the minimum of such a function can only be attained at a breakpoint. The most straightforward approach is therefore to enumerate all breakpoints $\{x_1, \cdots, x_k, x_{k+1} + m, \cdots, x_n + m\}$ and take the one that minimizes $g(\ell)$, which has a complexity of $\mathcal{O}(n^2)$. However, we can simplify further. First, taking the derivative, we get

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

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

However, if we replace $\#\{\ell > x_i\}$ by $\#\{\ell \geq x_i\}$ (the derivative at a discontinuous breakpoint can be viewed as arbitrary, so such an adjustment is permitted), then the condition $g'(\ell) = 0$ precisely means that "the number of breakpoints less than or equal to ℓ is exactly $n - k$." If we further assume that all breakpoints are pairwise distinct, then ℓ is exactly the $(n - k)$-th smallest element among all breakpoints when sorted in ascending order! Therefore, in the L1 case, a single sort suffices to obtain ℓ, which is quite elegant.

6 Reference Implementation

Reference implementations of the two 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))
```

A brief remark: in practical scenarios, the positive classes usually do not simply occupy the first k positions; their number and locations may vary. Therefore, in the above implementations, we use a mask vector to indicate the positive and negative classes.

The current implementation of `l2mep` only supports 1d inputs. To support batch dimensions, simply wrap it with `jax.vmap`. This algorithm finds the interval containing the zero by scanning all intervals; each step has complexity $\mathcal{O}(n)$, giving a total complexity of $\mathcal{O}(n^2)$. For improved

efficiency, one can switch to a binary search to locate the interval containing the zero, reducing the total complexity to $\mathcal{O}(n \log n)$.

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

7 Summary

This article introduced the operation of "Margin-Enforcing Projection (MEP)": given a score vector, it is projected onto the nearest vector satisfying the constraint that "the minimum of the positive-class scores exceeds the maximum of the negative-class scores by at least m." This can provide some new perspectives for traditional margin learning.