Optimization for Denoised Emotional Signal Extraction

Radim Nedbal

Overview

We formulate denoised emotional-signal extraction as an optimization over a mixed discrete–continuous parameter space Θ that parameterizes a p-dimensional submanifold of a Banach space $\mathcal{M}$ of maps. Each parameter choice $\theta \in \Theta$ defines a composed encoder

$$\Phi_\theta = \Psi_\theta \circ \Omega_\theta \circ \mathcal{E}_\theta,$$

mapping raw heatmap series $X_i(t)$ to denoised outputs $z_i^\theta(t) \in \mathbb{R}^L$. Categorical design choices such as wavelet family or embedding channel type index continuous submanifolds, so

$$\Theta = \bigcup_{c \in \mathcal{C}} \Theta_c.$$

Model and notation

- **Raw data** $X_i(t)$ — heatmap series for recording i, $t = 1, \dots, T$.
- **Time vector map** $\mathcal{E}_\theta$ — per-frame preprocessing and channel embedding (categorical choices: percentiles, bin heights, eigencfaces, etc.).
- **Temporal wavelet operator** Ω_θ — 1-D wavelet transform along time per channel (categorical choices: wavelet family, levels, padding).
- **Sequencewise encoder** Ψ_θ — estimator of latent source trajectories under a nonlinear-ICA model.

- **Branches** $c \in \mathcal{C}$ — categorical configurations; each branch has continuous parameter manifold Θ_c.
- **Encoder output** $z_i^\theta(t) = \Phi_\theta(X_i)(t) \in \mathbb{R}^L$.

Objectives

We support two objective families for learning θ (choose one or combine them with weights).

Lagged cross correlation objective

- **Per-pair normalized lag correlation** for scalar sequences $a(t), b(t)$ and lags $\ell \in [-L_{\max}, L_{\max}]$:

$$\rho_{ab}(\ell) = \frac{\sum_t a(t)\, b(t+\ell)}{\sqrt{\sum_t a(t)^2}\sqrt{\sum_t b(t+\ell)^2}}.$$

- **Differentiable surrogate** (soft-max over lags with temperature $\beta > 0$):

$$\tilde{\rho}_{ab} = \frac{1}{\beta} \log \sum_\ell \exp(\beta\, \rho_{ab}(\ell)).$$

- **Objective to maximize**

$$\mathcal{J}_{\text{corr}}(\theta) = \frac{1}{|\mathcal{C}|} \sum_{(i,j)\in\mathcal{C}} \tilde{\rho}_{z_i^\theta,\, z_j^\theta} \; - \; \lambda R(\theta),$$

where $\mathcal{C}$ is the set of within-segment pairs and $R(\theta)$ is a regularizer.

DTW classification objective

- **Soft-DTW distance**

$$d_{ij} = d_{\mathrm{sDTW}}(z_i^\theta, z_j^\theta; \gamma),$$

with smoothing $\gamma > 0$.

- **Pair sets:** positive pairs $\mathcal{P}$; negative pairs $\mathcal{N}$.

- **Logistic model (probabilistic)**

$$p_{ij}(\theta) = \sigma\Big(\alpha - w\frac{d_{ij}}{\tau}\Big), \qquad \mathcal{L}_{\mathrm{CE}}(\theta) = -\sum_{(i,j)} [y_{ij} \log p_{ij} + (1 - y_{ij}) \log(1 - p_{ij})],$$

where $y_{ij} \in \{0, 1\}$ and $\sigma(x) = (1 + e^{-x})^{-1}$.

- **Contrastive margin loss (geometric)**

$$\mathcal{L}_{\mathrm{ctr}}(\theta) = \sum_{(i,j)\in\mathcal{P}} d_{ij} + \mu \sum_{(i,j)\in\mathcal{N}} \big[m - d_{ij}\big]_+ + \lambda_{\mathrm{reg}} R(\theta),$$

with $[x]_+ = \max(0, x)$.

- **Combined objective**

$$\mathcal{L}(\theta) = \lambda_{\mathrm{CE}} \mathcal{L}_{\mathrm{CE}}(\theta) + \lambda_{\mathrm{ctr}} \mathcal{L}_{\mathrm{ctr}}(\theta) + \lambda_{\mathrm{reg}} R(\theta).$$

Tune $\lambda_{\mathrm{CE}}, \lambda_{\mathrm{ctr}}, \lambda_{\mathrm{reg}}$ on validation data.

Training and optimization

Mixed discrete–continuous search

- **Search space:** treat categorical axes as discrete branches and write the global space as a disjoint, tagged union

$$\Theta = \bigsqcup_{c \in \mathcal{C}} \{c\} \times \Theta_c,$$

so each candidate is a pair (c, ϕ) with $\phi \in \Theta_c$.

- **Two-stage search**
  - **Coarse screening (parallel, low fidelity):** randomized trials across categorical choices with short training, downsampled data, or smaller models. Use Hyperband / successive halving to allocate budget adaptively. Record cheap validation metrics and complexity proxies (params, FLOPs, latency).
  - **Focused optimization (per branch):** for top-k branches run full gradient-based optimization on θ_c; tune hyperparameters with TPE (Optuna `TPESampler`) or Bayesian optimization; early stop on held-out segments; optionally warm-start from coarse runs.
- **Optional joint relaxation:** use Gumbel-Softmax / Concrete relaxations (straight-through variant recommended) during coarse training; discretize and re-evaluate selected branches.

Within-branch optimization

- **Autodiff:** compute gradients through Φ_θ and the chosen differentiable objective (soft-lag or soft-DTW).
- **Manifold constraints:** if Θ_c has constraints, use Riemannian updates (project Euclidean gradient to tangent space, retract) or parametrize constrained variables.
- **Positivity constraints:** parametrize positive scalars as exponentials (e.g., $w = \exp(\eta)$, $\tau = \exp(\zeta)$).

Batching and negative mining

- **Balanced minibatches:** ensure multiple positives per anchor and controlled negatives; denote $\mathcal{P}_{\text{batch}}, \mathcal{N}_{\text{batch}}$.
- **Semi-hard negative mining (default):** select negatives satisfying $d(a,p) < d(a,n) < d(a,p) + m$; if none, use hardest in-batch negative. This yields informative gradients while avoiding extreme noisy outliers.
- **Scaling DTW:** reduce $O(T^2)$ cost via downsampling, windowed DTW (Sakoe–Chiba), low-dim projections (PCA), or caching repeated distances.

Regularization, safeguards, and evaluation

Regularizers and safeguards

- **Energy constraint** (prevent collapse):

$$\frac{1}{T}\sum_{t=1}^{T} \|z^{\theta}(t)\|^2 \geq \varepsilon,$$

or add soft penalty $\lambda_{\text{energy}} \max(0, \varepsilon - \frac{1}{T}\sum_t \|z^{\theta}(t)\|^2)$.

- **Temporal penalties:** ℓ_2 smoothness $\sum_t \|z(t+1) - z(t)\|^2$; optional ℓ_1 on first differences for sparse transients.
- **Complexity penalty:** $C(c, \theta_c)$ to penalize FLOPs, parameter count, long filters.
- **Validation and early stopping:** use held-out segments and cross-subject splits; monitor validation gap.
- **Robustness checks:** sensitivity to padding, filter length, and categorical axes; run single-axis ablations.

Evaluation protocol

- **Denoising:** Signal-to-noise ratio (SNR), Mean squared error (MSE) on emotion-relevant bands (use synthetic injections if no ground truth).
- **Alignment:** average soft-lag correlation $\tilde{\rho}$ for within- vs between-segment pairs.
- **Pairwise classification:** accuracy, ROC AUC, precision/recall on held-out pairs; calibration for logistic models.
- **Event detection:** precision/recall and temporal localization error for transients.
- **Generalization:** cross-session and cross-subject performance; report validation gap and complexity vs performance.
- **Ablations:** effect of categorical choices, regularizers, smoothing γ, and temperatures (β, τ).

Implementation pseudocode

```
# Stage A: coarse screening (parallel)
for c in categorical_configs:            # parallelizable
    for r in range(N_random_inits):
        theta = sample_random_init(c)
        train_short(theta, data_downsampled)   # few epochs, small model
        val_metric = evaluate(theta, val_set_small)
        record_result(c, theta, val_metric)
C_top = select_top_k_configs()

# Stage B: focused optimization (per selected branch)
for c in C_top:
    theta = initialize_theta(c)           # optional warmstart
    for epoch in range(1, N_epochs+1):
        for batch in data_loader:
            X = batch.recordings
            Z = Phi_theta(X)              # forward: E_theta, Omega_theta, Psi_theta
            P_batch, N_batch = sample_pairs(batch, strategy='balanced')
            if objective == 'lagged_corr':
                rho_tilde = compute_soft_lag(Z, P_batch, beta)
```

```
            L_obj = -rho_tilde.mean() + lambda_reg * R(theta)
        elif objective == 'dtw':
            D_pos = soft_dtw_pairwise(Z, P_batch, gamma)
            D_neg = soft_dtw_pairwise(Z, N_batch, gamma)
            L_ctr = D_pos.sum() + mu * torch.relu(m - D_neg).sum()
            logits = alpha - w * torch.cat([D_pos, D_neg]) / tau
            L_ce = cross_entropy(torch.sigmoid(logits), labels_for_pairs)
            L_obj = lambda_ce * L_ce + lambda_ctr * L_ctr + lambda_reg * R(theta)
        L_obj.backward()              # autodiff through soft-DTW / soft-lag -> Phi_theta
        if manifold_constraints:
            g = get_euclidean_grad(theta)
            g_tangent = project_to_tangent(g, theta)
            theta = retraction_step(theta, g_tangent, optimizer)
        else:
            optimizer.step()
            optimizer.zero_grad()
    if early_stop_condition(evaluate(theta, val_set)):
        break
save_checkpoint(theta, c)
```