Ideation

LTS2 Bois

November 28, 2019

0 Step 0: Goodness of Cotan

Assess how well both eigenvectors and eigenvalues behave with cotan formula and mass matrix.

Should pay attention to transferability at this early stage. Different graphs based on GHCN.

- Build graph from all stations at a particular day(somewhere near 12k).
- Build graph from all persistent stations throughout a specified time window.
  For example:
  - Around 9000 if the window is taken as 30 days.
  - Around 5500 if the window is taken as a year.

Ideally, these graphs should have similar spectra.

1 Step 1: GHCN Interpolation

1.1 Definitions and Notations

Definition 1. *At date t, the set of all stations is denoted as S_t*

Definition 2. *Given time window $[T_1, T_2]$, the persistent set of stations throughout time window $[T_1, t_2]$, denoted as $\Pi_{[T_1,T_2]}$, is defined as*

$$\Pi_{[T_1,T_2]} := \bigcap_{t=T_1}^{T_2} S_t$$

It should be noted that given a time window $[T_1, T_2]$, the notion of persistent set of stations introduces a natural partition of S_t for any $t \in [T_1, T_2]$.

Definition 3. *Given time window $[T_1, T_2]$, the super set of stations throughout time window $[T_1, T_2]$, denoted as $\Omega_{[T_1,T_2]}$, is defined as*

$$\Omega_{[T_1,T_2]} := \bigcup_{t=T_1}^{T_2} S_t$$

1.2 Graph Construction

Let $\mathcal{S}_n$ be the set of all sets of stations with cardinality n. And let

$$\mathrm{MESH}_n : \mathcal{S}_n \to \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$$

be the map from point clouds of stations to Laplacian matrices of the graph constructed by convex hull. And

$$\mathrm{kNN}_n : \mathcal{S}_n \to \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$$

is defined in exactly the same way.

Then we need to define the masking operation, which is effectively cutting off the outgoing edges from a station while keeping the ingoing edges.

$$M_j : \mathbb{R}^{\mathrm{n}\times\mathrm{n}} \to \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$$

which works by setting $L_{ij} \leftarrow 0 \ \ \forall\, i \neq j$

Let $\tilde{L} := M_j(L)$, then

$$(\tilde{L}x)_i = \sum_{k=1}^{n} \tilde{L}_{ik}x_k = \sum_{k\neq j} L_{ik}x_k, \quad \forall\, i \neq j$$

effectively removing the dependency on the knowledge of x_j, which we won't know when doing interpolation anyway.

The graph construction given a time window $[T_1, T_2]$ then works as following:

- Let $\Pi_{[T_1,T_2]} \subset \omega \subset \Omega_{[T_1,T_2]}$. Preferably randomized.
- $L = \mathrm{MESH}(w)$.
- Let $\pi \subset \Pi_{[T_1,T_2]}$. Preferably randomized.
- For each $s \in \omega\backslash\pi$, $L \leftarrow M_s(L)$.

By construction, for each $s \in S_t\backslash\pi$, we actually have the ground truth data at s. Nonetheless, we'd like to pretend that we don't know the data during training in order to compute training loss, just like standard supervised regression tasks.

2 Graph Convolution

In this section we take the graph Laplacian L as given. The convolution is carried out in almost the same way as described in [1], with some small but crucial twists.

If the parameterized monomial filter is used, according to [1] the filtering can be expressed as

$$g_\theta(L) := \sum_{k=0}^{K-1} \theta_k L^k$$

while the parameterized Chebyshev filter can be written as

$$g_\theta(L) := \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})$$

where

- we first need to rescale L, $\tilde{L} := 2L/\lambda_{\max} - I$. By doing this, the eigenvalues of L are all in the range [-1, 1], making the Chebyshev polynomial well defined.
- we then need to compute $T_k(L)x$ recursively. Namely,

$$T_0(\tilde{L})x = x, \;\; T_1(\tilde{L})x = \tilde{L}x, \;\; T_k(\tilde{L})x = 2\tilde{L}\, T_{k-1}(\tilde{L})x - T_{k-2}(\tilde{L})x$$

Monomial filter faces the problem of unbounded eigenvalues as K grows large. In the meantime, the Chebyshev filter faces the problem of scaling the Laplacian incorrectly with different sampling size. We'll further discuss this difficulty later.

The first important modification is to change the summation $\sum_{k=0}^{K-1}$ into $\sum_{k=1}^{K}$. By doing so, we make sure that output from the first convolution layer doesn't depend on the artificially hidden data. Together with masking Laplacian, we make sure the output from future layers won't depend on the hidden data either.

3 Training and Testing

3.1 Training

Let $\mathcal{T}$ be a set of time windows (not necessarily of the same length, not necessarily disjoint). Let **SAMPLE**ω, **SAMPLE**π be two randomized subprocedure that generate stream of ω and π.
Let d be the number of features used. Let $x_t : \omega \to \mathbb{R}^{\mathrm{d}}$ be understood as the input feature function defined on the superset ω, defined as

$$x_t(s) = \begin{cases} \text{feature set available at } s & \text{if } s \in S_t \\ 0 & \text{otherwise} \end{cases}$$

Note that x_t can also be viewed as a vector in $\mathbb{R}^{\mathrm{d}\times|\omega|}$

The pseudocode is as follows

Algorithm 1 GHCN Interpolation: Training

procedure TRAINING
  for $[T_1, T_2] \in \mathcal{T}$ **do**
    Compute Π, Ω of $[T_1, T_2]$
    for $\Pi \subset \omega \subset \Omega \leftarrow$ **SAMPLE**ω **do**
      L = MESH(ω)
      for $\pi \subset \Pi \leftarrow$ **SAMPLE**π **do**
        $\bar{L} \leftarrow M_{\omega\backslash\pi}(L)$ ▷ shorthand for masking everything in the set $\omega\backslash\pi$
        GCN $\leftarrow$ GCN($\bar{L}$)
        for $t \in [T_1, T_2]$ **do**
          $y = \mathrm{GCN}(x_t) \in \mathbb{R}^{|\omega|}$ ▷ $y \in \mathbb{R}^{|\omega|}$ is output of Dense Regression
          loss := $\sum_{s\in S_t\backslash\pi}[y(s) - y^*(s)]^2$ ▷ Loss is computed only on $S_t\backslash\pi$
          loss.backprop()
        end for
      end for
    end for
  end for
end procedure

References

[1] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.