Problem Set 2

6.S091: Topics in Causality
IAP 2024

Problem 1: Implementing DAGs with NO TEARS [30 points]

Preliminaries [10 point]

(a) We have

$$\begin{aligned}\operatorname{tr}(\mathcal{A}(W)^k) &= \sum_{i=1}^{d}(\mathcal{A}(W)^k)_{ii} \\ &= \sum_{i=1}^{d}\sum_{t_1,\ldots,t_{k-2}=1}^{d}\mathcal{A}(W)_{it_1}\mathcal{A}(W)_{t_1t_2}\ldots\mathcal{A}(W)_{t_{k-2}i}.\end{aligned}$$

Note that if $i - t_1 - ... - t_{k-2} - i$ is a (length-k) cycle in $\mathcal{G}(W)$,

$$\mathcal{A}(W)_{it_1}\mathcal{A}(W)_{t_1t_2}\ldots\mathcal{A}(W)_{t_{k-2}i} = 1,$$

otherwise

$$\mathcal{A}(W)_{it_1}\mathcal{A}(W)_{t_1t_2}\ldots\mathcal{A}(W)_{t_{k-2}i} = 0.$$

As the sums in $\operatorname{tr}(\mathcal{A}(W)^k))$ consist of different and all sequences of $i, t_1, ..., t_{k-2}, i$, it equals to the number of length-k cycles in $\mathsf{G}(W)$.

(b) By the definition of matrix exponential and the linearity of trace, we have

$$\operatorname{tr}(e^{W\circ W}) = \operatorname{tr}(I) + \sum_{k=1}^{+\infty}\frac{1}{k!}\cdot\operatorname{tr}((W\circ W)^k),$$

where I is the $d\times d$ identity matrix. Since all entries of $W\circ W$ are non-negative and $\operatorname{tr}(I) = d$, we immediately have $\operatorname{tr}(e^{W\circ W}) \geq d$, i.e., $h(W) \geq 0$.

When $h(W) = 0$, there is $\operatorname{tr}((W\circ W)^k) = 0$ for all positive integer k. Note that using a similar proof as **(a)**, we have that $\operatorname{tr}((W\circ W)^k)$ equals to the weighted sum of length-k circles in $\mathsf{G}(W)$, where the weight of each circle is the square of the product of weights of the edges on this circle. When a circle is present, this weight is greater than 0. Therefore $\operatorname{tr}((W\circ W)^k) = 0$ if and only there is no length-k circle in $\mathsf{G}(W)$. As this holds for all positive integer k, we know that $h(W) = 0$ if and only if there is no cycles in $\mathsf{G}(W)$.

The gradient of $h(W)$ is $\nabla h(W) = e^{W\circ W}\circ 2W$.

Implementing the constraint [10 point]

(c) See the following snippet.

```
import numpy as np
import scipy.linalg as slin

def h(W):
        E = slin.expm(W * W)
```

```
        h = np.trace(E) - len(W)
        return h
```

(d) $h(W_1) = 0$, $h(W_2) = 0.2826$.

Optimizing with augmented Lagrangian [10 point]

(e) See the following snippet.

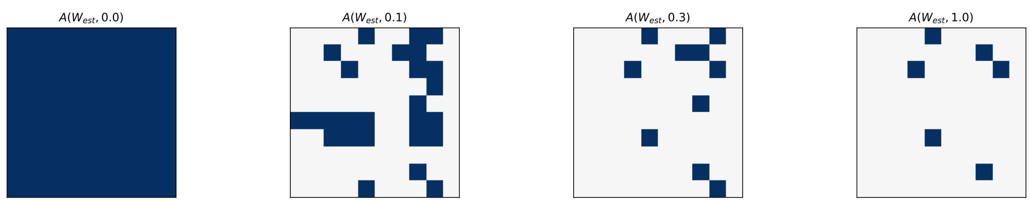
```
def notears_linear(X, max_iter=100, h_tol=1e-6, rho_max=1e+16):
    """Solve min_W F(W) s.t. h(W) = 0 using augmented Lagrangian.

    Args:
        X (np.ndarray): [n, d] sample matrix
        max_iter (int): max num of dual ascent steps
        h_tol (float): exit if |h(W_est)| <= htol
        rho_max (float): exit if rho >= rho_max

    Returns:
        W_est (np.ndarray): [d, d] estimated weights
    """
    def _loss(W):
        """Evaluate value and gradient of loss."""
        W = W.reshape((d, d))
        M = X @ W
        R = X - M
        loss = 0.5 / X.shape[0] * (R ** 2).sum()
        G_loss = - 1.0 / X.shape[0] * X.T @ R
        return loss, G_loss

    def _h(W):
        """Evaluate value and gradient of acyclicity constraint."""
        W = W.reshape((d, d))
        E = slin.expm(W * W)  # (Zheng et al. 2018)
        h = np.trace(E) - d
        G_h = E.T * W * 2
        return h, G_h

    def _func(W):
        """Evaluate value and gradient of augmented Lagrangian."""
        loss, G_loss = _loss(W)
        h, G_h = _h(W)
        obj = loss + 0.5 * rho * h * h + alpha * h
        g_obj = np.concatenate((G_loss + (rho * h + alpha) * G_h), axis=None)

        return obj, g_obj

    n, d = X.shape
    W_est, rho, alpha, h = np.zeros((d*d)), 1.0, 0.0, np.inf

    for _ in range(max_iter):
        W_new, h_new = None, None

        W_new = sopt.minimize(_func, W_est, method='L-BFGS-B', jac=True).x
        h_new, _ = _h(W_new)

        alpha += rho * h_new

        if h_new > h / 2:
```

```python
            rho *= 10

        W_est, h = W_new, h_new

        if h <= h_tol or rho >= rho_max:
            break

    return W_est.reshape((d, d))
```

(f) See the following for one run of the above algorithm, compared with the ground-truth W.

(a) Ground-truth W (b) Estimated W_{est}

Figure 1: One run of the above algorithm v.s. the ground-truth W.



(a) $\omega = 0.0$ (b) $\omega = 0.1$ (c) $\omega = 0.3$ (d) $\omega = 1.0$

Figure 2: Adjacency matrix $\mathcal{A}(W_{est}, \omega)$ with different threshold values.