Controlling overlap

November 22, 2020

1 Controlling the overlap of first-layer weights in 2LNN in the meanfield regime

1.1 Nov 2020

[10]:
```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'

import numpy as np
import numpy.random as rnd

import matplotlib as mpl
import matplotlib.pyplot as plt

plt.rc("font", **{"size": 14})
plt.rc("lines", linewidth=2.0)
mpl.rcParams['lines.markersize'] = 12
mpl.rcParams['lines.markeredgewidth'] = 2
```

We propose a measure for the overlap of the first layer weights of two-layer neural nets $\phi_\theta : \mathbb{R}^N \to \mathbb{R}$ in the meanfield regime:

$$\phi_\theta(x) = \frac{1}{K} \sum_k^K v^k g(w^k x) \tag{1}$$

where θ denotes all the trainable parameters of the model.

We first draw a weight matrix for the first teacher $W_1 = (w_1^k) \in \mathbb{R}^{N \times K}$ element-wise i.i.d. from the normal distribution and orthonormalise it. The first-layer weights for the second teacher W_2 are obtained from

$$W_2 = \alpha W_1 + (1 - \alpha) Z \tag{2}$$

where Z is another matrix with i.i.d. Gaussian elements and $0 < \alpha < 1$ is an interpolation parameter. We now compute the overlap of the two weight matrices for different values of α.

[11]:
```
N = 15  # input dimension
K = 1000  # num of neurons in the hidden layer
```

[12]:
```
alphas = np.linspace(0, 1)  # interpolation parameter
```

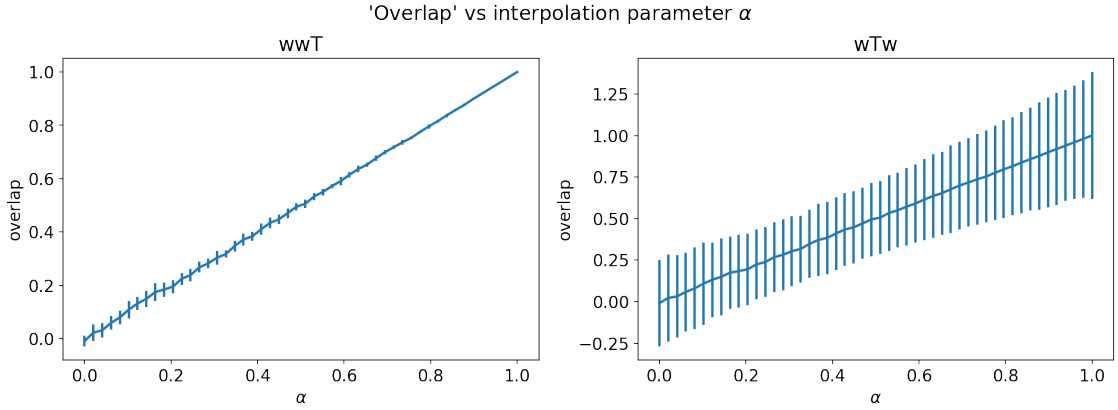
We will calculate the mean and standard deviation of two overlap matrices:

$$\frac{WW^\top}{K} \qquad \text{and} \qquad \frac{W^\top W}{N} \tag{3}$$

[13]:
```
means = {"wwT": [None] * len(alphas), "wTw": [None] * len(alphas)}
stds = {"wwT": [None] * len(alphas), "wTw": [None] * len(alphas)}
```

[14]:
```
for idx, alpha in enumerate(alphas):
    ws = [rnd.randn(N, K), None]  # first layer of weights

    # orthonormalise the weights of the first teacher;
    # this reduces the fluctuations in the overlap measured below
    overlap = ws[0] @ ws[0].T / K
    L = np.linalg.cholesky(overlap)
    ws[0] = np.linalg.inv(L) @ ws[0]

    # Compute the first-layer weights of the second teacher
    ws[1] = alpha * ws[0]  + (1 - alpha) * rnd.randn(*ws[0].shape)

    # calculate diagonal elements of overlaps
    overlaps = {"wwT": np.diag(ws[0] @ ws[1].T / K),
                "wTw": np.diag(ws[0].T @ ws[1] / N)}
    for key in ["wwT", "wTw"]:
        means[key][idx] = np.mean(overlaps[key])
        stds[key][idx] =  np.std(overlaps[key])
```

[19]:
```
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(4/3 * 2 * 5, 5))

plt.suptitle(r"'Overlap' vs interpolation parameter $\alpha$")
for idx, key in enumerate(["wwT", "wTw"]):
    ax[idx].set_title(key)
    ax[idx].errorbar(alphas, means[key], yerr=stds[key], label=key)
    ax[idx].set_xlabel(r"$\alpha$")
    ax[idx].set_ylabel("overlap")
plt.tight_layout(rect=[0,0,1,.95])
plt.show()
```



Note that orthonormalising the weights of the first teacher (interpreted as vectors in K dimensions) increases the fluctuations in $W^\top W$, but these fluctuations are in any case much larger than those in $WW^\top$.