Network expansions modify loss landscape: the case of quadratic functions

In the previous sections, I have examined how the geometry of the loss landscape affects learning performance. I now present how a change in network architecture modifies the loss landscape and hence the learning performance.

In this subsection we consider the case of a quadratic loss function as it is mathematically tractable. This example illustrates how adding parameters to a function changes the geometry of the loss landscape in a way that can either be beneficial or detrimental to learning.

1D quadratic function

First, consider the simplest example possible. Let $f : \mathbb{R} \to \mathbb{R}$ be a one dimensional quadratic function

$$f(w) = w^2 \tag{1}$$

This function defines a landscape in 2D that is just a parabola with minimum at $x = 0$ and upward curvature. The slope at some point w_0 is given by

$$\left.\frac{df(w)}{dw}\right|_{w=w_0} = 2w_0 \tag{2}$$

it increases as it gets away from the origin. On the other hand, the curvature is constant in all space (this property holds for all quadratic functions). The curvature is given by the second derivative.

$$\left.\frac{d^2 f(w)}{dw^2}\right|_{w=w_0} = 2 \tag{3}$$

From the previous section we saw that the slope and the curvature control learning performance. In particular, the magnitude of the gradient at some point, the average curvature and the spread of the eigenvalues of the Hessian. In this case, the magnitude of the gradient is just the $2|w_0|$, the average curvature and the spread of the eigenvalues are equal to the only eigenvalue $\lambda_1 = 2$.

If some iterative first-order learning algorithm is performed on this function it will descend the parabola with some learning speed and will reach some steady state value determined by the learning algorithm and the learning step. In particular gradient descent with a learning step $\gamma < 1$ will converge to the minimum $w = 0$. Figure **??** A) shows the loss landscape and the trajectory given by gradient descent with learning step $\gamma = 0.01$ and initial point $w = 10$.

We consider a more biologically relevant learning rule: gradient descent with noise. We assume the weight update is given by

$$w(t+1) = w(t) - \gamma \left.\frac{df(w)}{dw}\right|_{w=w(t)} + \eta\epsilon(t) \tag{4}$$

where $w(t+1)$ is the weight at step $t+1$, γ is the learning step, η is the amount of noise and $\epsilon(t)$ is the noise which is drawn from a random normal distribution $\mathcal{N}(0,1)$. In this case, the steady state value will be non-zero for non-zero γ (see Figure **??** B)).

We can compute the expected instantaneous learning speed ν_t over the distribution of the noise ϵ

$$\mathbb{E}[\nu_t] = \mathbb{E}[-\frac{\Delta f(w_t)}{\delta t}] \tag{5}$$

$$= \mathbb{E}[-\frac{1}{1}(\delta w_t \frac{df(w_t)}{dw} + \delta w_t \frac{d^2 f(w_t)}{dw^2}\delta w_t] \tag{6}$$

$$= \gamma \left(\frac{df(w_t)}{dw}\right)^2 - \gamma^2 \frac{df(w_t)}{dw}\frac{d^2 f(w_t)}{dw^2}\frac{df(w_t)}{dw} - \eta^2 \mathbb{E}[\epsilon_t \frac{d^2 f(w_t)}{dw^2}\epsilon_t] \tag{7}$$

$$= \gamma \left(2w_t\right)^2 - \gamma^2 (2w_t)^2 \cdot 2 - \eta^2 \mathbb{E}[\epsilon_t 2\epsilon_t] \tag{8}$$

$$= \gamma \left(2w_t\right)^2 - 2\gamma^2 (2w_t)^2 - 2\eta^2 \mathbb{E}[(\epsilon_t)^2] \tag{9}$$

$$= 4\gamma \left(w_t\right)^2 - 8\gamma^2 (w_t)^2 - 2\eta^2 \mathbb{E}[(\epsilon_t)^2] \tag{10}$$

$$= 4\gamma \left(w_t\right)^2 - 8\gamma^2 (w_t)^2 - 2\eta^2 \tag{11}$$

where we have used $\mathbb{E}[\epsilon_t^2] = Var(\epsilon_t) = 1$. The learning noise $\eta\epsilon_t$ decreases the learning speed proportionally to η and the average curvature. It is often the case that the learning noise is proportional to the learning step γ. Indeed, if the plasticity in the "task-relevant direction" increases, inevitably the noise or plasticity in "task-irrelevant direction" also increases. If we fix the signal to noise ratio $\sigma = \frac{\gamma}{\eta}$, then the learning speed is

$$\mathbb{E}[\nu_t] = 4\gamma \left(w_t\right)^2 - 8\gamma^2 (w_t)^2 - 2\left(\frac{\gamma}{\sigma}\right)^2 \tag{12}$$

For a given weight point w_t, the learning speed increases with the learning step until it reaches a maximum for

$$\gamma^* = \frac{(w_t)^2}{4(w_t)^2 + 1/\sigma^2} \tag{13}$$

The learning speed increases with w_t (i.e. when the weight point is further away from the minimum of the loss function). Indeed, the further away, the larger the slope so the bigger the learning steps can be.

Furthermore, we can compute the expected local task difficulty which relates to

the steady state loss.

$$\mathbb{E}[G_t] = \gamma \frac{1}{\left(\frac{df(w_t)}{dw}\right)^2} \frac{df(w_t)}{dw} \frac{d^2 f(w_t)}{dw^2} \frac{df(w_t)}{dw} + \frac{\eta^2}{\gamma} \frac{1}{\left(\frac{df(w_t)}{dw}\right)^2} \mathbb{E}[\epsilon_t \frac{d^2 f(w_t)}{dw^2} \epsilon_t] \quad (14)$$

$$= 2\gamma + \frac{\eta^2}{\gamma} \frac{2}{(2w_t)^2} \mathbb{E}[(\epsilon_t)^2] \quad (15)$$

$$= 2\gamma + \frac{\eta^2}{\gamma} \frac{2}{(2w_t)^2} \mathbb{E}[(\epsilon_t)^2] \quad (16)$$

$$= 2\gamma + \frac{\eta^2}{\gamma} \frac{2}{(2w_t)^2} \quad (17)$$

Note that the local task difficulty is only relevant near steady state —when w_t is close to the minimum 0. As opposed to the learning speed ν_t is mainly relevant at initial stages of learning, when $|w_t| >> 0$. So in general, the relevant values of $\mathbb{E}[\nu_t]$ and $\mathbb{E}[G_t]$, to determine learning speed and steady state loss, are computed be for different t and hence different w_t.

For fixed σ we get

$$\mathbb{E}[G_t] = 2\gamma + \frac{\gamma}{\sigma^2} \frac{2}{(2w_t)^2} = 2\gamma \left(1 + \frac{1}{\sigma^2 (2w_t)^2}\right) \quad (18)$$

the local task difficulty increases linearly with the learning step γ. Note that, the local task difficulty is minimal (i.e. equal to zero) for zero learning step γ. However, if the learning step were zero, there would be no learning and the steady state loss would not be minimal. The local task difficulty represents the effect of the curvature of the loss landscape on learning near steady state. This quantity is related to the steady state loss in the sense that if the local task difficulty decreases with a change in the network architecture, then the optimal steady state loss decreases as well.

From this example of a 1D loss function with a noisy gradient descent learning rule, we have seen that the learning speed increases with the learning step until it reaches a optimal value γ^* determined by the geometry of the function and the quality of learning rule determined σ. Because of the noise, this kind of learning rule doesn't converge to the exact minimum 0; the steady state loss increases with the learning step γ and decreases with σ. Indeed, it is the learning rule noise that impairs steady state performance.

If f is the loss function for training a network with a single parameter w, how does the loss function change if we add a second parameter to the network? This would correspond to a family of network expansions $\phi_a : \mathbb{R} \to \mathbb{R}^2$ such that $\phi_a(w) = (w, a)$ and the inverse mapping is defined $\phi^{-1}((w_1, w_2)) = w_1$. The network expansion leads to a new loss function in 2D space. The exact form of this loss function is determined by the network architecture and the nature of the learning problem —how the loss function is defined.

2D quadratic functions

Abstractly, there are infinitely many 2D functions. However, there will generally be constrains on the possible 2D functions that can arise from a network expansion. Assume that the 2D functions must be quadratic and have the same average curvature $\frac{Tr(H)}{N}$ as the 1D function. We consider two different quadratic functions that satisfy both of these constraints.

First, we consider the 2D quadratic function $f_{21} : \mathbb{R}^2 \to \mathbb{R}$ with

$$f_{21}(w_1, w_2) = 2w_1^2 \tag{19}$$

where w_1 and w_2 are the variables. In 3D landscape, this quadratic bivariate function has the form of a valley with minimum for $w_1 = 0$. Indeed, it is essentially a parabola translated through the w_2 axis. The gradient of this function is the vector

$$\nabla f_{21}(w_1, w_2) = \begin{pmatrix} 4w_1 \\ 0 \end{pmatrix} \tag{20}$$

meaning that the slope in the w_2 direction is zero and in the w_1 direction it is proportional to w_1. The hessian matrix is constant in $\mathbb{R}$ and given by

$$\nabla^2 f_{21}(w_1, w_2) = \begin{pmatrix} 4 & 0 \\ 0 & 0 \end{pmatrix} \tag{21}$$

This matrix has two eigenvectors $(1 \quad 0)^T$ and $(0 \quad 1)^T$ with corresponding eigenvalues $\lambda_1 = 4$ and $\lambda_2 = 0$. Hence the curvature is zero in the w_2 direction and 4 in the w_1 direction. Furthermore, the average curvature given by the average eigenvalue is

$$\frac{Tr(\nabla^2 f_{21})}{N} = \frac{4}{2} = 2 \tag{22}$$

where $N = 2$ is the number of variables. The spread of the eigenvalues

$$\frac{Tr((\nabla^2 f_{21})^3)}{Tr\left((\nabla^2 f_{21})^2\right)} = 4 \tag{23}$$

This 2D quadratic function has a larger slope $||\nabla f_{21}(\phi_a(w))|| = 4w \geq \frac{df(w)}{dw} = 2|w|$, same average curvature but larger spread of eigenvalues compared to the 1D function.

Second, we consider the 2D quadratic function $f_{22} : \mathbb{R}^2 \to \mathbb{R}$ with

$$f_{22}(w_1, w_2) = w_1^2 + w_2^2 \tag{24}$$

where w_1 and w_2 are the variables. In 3D landscape, this quadratic bivariate function has the form of a parabola with minimum for $w_1 = 0, w_2 = 0$. The gradient of this function is the vector

$$\nabla f_{22}(w_1, w_2) = \begin{pmatrix} 2w_1 \\ 2w_2 \end{pmatrix} \tag{25}$$

meaning that the slope in the w_1 and w_2 directions for $w_1 = w_2$ are equal. The hessian matrix is constant in $\mathbb{R}$ and given by

$$\nabla^2 f_{22}(w_1, w_2) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \tag{26}$$

This matrix has two eigenvectors $(1 \quad 0)^T$ and $(0 \quad 1)^T$ with corresponding eigenvalues $\lambda_1 = 2$ and $\lambda_2 = 2$. The curvature is equal in all directions, indeed, the loss landscape is symmetrical. As expected, the average curvature given by the average eigenvalue is

$$\frac{Tr(\nabla^2 f_{22})}{N} = \frac{4}{2} = 2 \tag{27}$$

where $N = 2$ is the number of variables. The spread of the eigenvalues

$$\frac{Tr((\nabla^2 f_{22})^3)}{Tr\left((\nabla^2 f_{22})^2\right)} = 2 \tag{28}$$

This 2D quadratic function has the same average curvature and spread of eigenvalues of the hessian than the 1D quadratic function. The slope $||\nabla f_{22}(\phi_a(w))|| = 2\sqrt{w^2 + a^2} = 2||\phi_a(w)||$ is equal to the slope of the 1D function as long as $||\phi_a(w)|| = |w|$ (i.e. as long as point in 2D space is the same distance as the point in 1D space).

We have considered two qualitative different expansions of the 1D quadratic function f. The first expansions adds a zero eigenvalue to the hessian (i.e. a direction of zero curvature), the second expansion adds an eigenvalue equal to the average eigenvalue. Figure **??** shows the loss landscape for the 1D and the two different 2D functions. We will show that these two different types of expansions have a different effect on learning performance. One, is beneficial —it speeds up learning—, the other doesn't.

$N-$dimensional quadratic functions

We can generalise these two types of functions to N dimensions. As above, we assume that all functions are quadratic and their hessian has the same average eigenvalue equal to ρ.

Single non-zero eigenvalue The first type of function is given by $f_{N1} : \mathbb{R}^N \to \mathbb{R}$ with

$$f_{N1}(\mathbf{w}) = \frac{\rho N}{2} w_1^2 \tag{29}$$

where $\mathbf{w} = (w_1, w_2, \cdots, w_N)^T \in \mathbb{R}^N$. The gradient of this function is the vector

$$\nabla f_{N1}(\mathbf{w}) = \begin{pmatrix} \rho N w_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{30}$$

meaning that the slope in the w_1 direction it is proportional to w_1 and is zero in all the other directions. The hessian matrix is constant in $\mathbb{R}$ and given by

$$H_{N1} \equiv \nabla^2 f_{N1}(\mathbf{w}) = \begin{pmatrix} \rho N & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & & 0 \end{pmatrix} \tag{31}$$

This matrix has N orthogonal eigenvectors given by the Euclidean unit vectors $e_i, i = 1, \cdots, N$ all the eigenvalues are zero except the first eigenvalue $\lambda_1 = \rho N$. Hence the curvature is zero in all directions except in the w_1 direction. As expected, the average curvature given by the average eigenvalue is

$$\frac{Tr(H_{N1})}{N} = \rho \tag{32}$$

The spread of the eigenvalues

$$\frac{Tr((H_{N1})^3)}{Tr\left((H_{N1})^2\right)} = \rho N \tag{33}$$

increases with the network size. The slope and the spread of the eigenvalues increases linearly with the number of variables N, whereas the average eigenvalue increases with network size.

How does adding parameters to the loss function affect learning performance?

Learning performance The learning rule is the same as in 1D, in vector form

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma \nabla f_{21}(\mathbf{w}_t) + \eta \epsilon(t) \tag{34}$$

where we have used the vector notation where $\mathbf{w} = (w_1, w_2, \cdots, w_N)^T$ and ϵ are vectors.

We can compute the expected learning speed and local task difficulty given the weight state $\mathbf{w}_t = \frac{w_0}{\sqrt{N}}(1, 1, \cdots, 1)^T$, this guarantees that each point in N-dimensional space is at the same distance from the minimum of the function $(0, 0, \cdots, 0)^T$. Furthermore, this choice guarantees that $f_{N1}(\mathbf{w}_t) = f_{N2}(\mathbf{w}_t) = \frac{\rho}{2} w_0^2$ for all N. As the learning speed is mainly relevant far from the minimum, we assume $|w_0| >> 0$. We can compute the learning speed

$$\mathbb{E}[\nu_t]_{N1} = \gamma ||\nabla f_{N1}(\mathbf{w}_t)||^2 - \gamma^2 \nabla f_{N1}(\mathbf{w}_t)^T H_{N1} \nabla f_{N1}(\mathbf{w}_t) - \eta^2 \mathbb{E}[\epsilon_t^T H_{N1} \epsilon_t] \tag{35}$$

$$= \gamma \rho^2 N^2 w_{1t}^2 - \gamma^2 (\rho N w_{1t})^2 \rho N - \eta^2 \frac{Tr(H_{N1})}{N} \mathbb{E}[||\epsilon_t||^2] \tag{36}$$

$$= \gamma \rho^2 N^2 w_{1t}^2 - \gamma^2 (\rho N w_{1t})^2 \rho N - \eta^2 \rho \mathbb{E}[||\epsilon_t||^2] \tag{37}$$

$$= \gamma \rho^2 N^2 w_{1t}^2 - \gamma^2 (\rho N w_{1t})^2 \rho N - \eta^2 \rho N \tag{38}$$

$$= \gamma \rho^2 N w_0^2 - \gamma^2 \rho^3 w_0^2 N^2 - \eta^2 \rho N \tag{39}$$

where we have used the fact that each element of the vector ϵ_t is drawn from a random normal distribution $\mathcal{N}(0,1)$ so the expected $\mathbb{E}[||\epsilon_t||^2] = N\mathbb{E}[\epsilon_t^2] = N$. For fixed σ, we get

$$\mathbb{E}[\nu_t]_{N1} = \gamma\rho^2 N w_0^2 - \gamma^2\rho^3 N^2 w_0^2 - \left(\frac{\gamma}{\sigma}\right)^2 \rho N \tag{40}$$

The learning speed $\mathbb{E}[\nu_t]_{N1}$ is a quadratic function in the learning step γ and in N. Consider N to be fixed, the learning speed increases with γ until it reaches a maximal value for γ^*_{N1}.

$$\gamma^*_{N1} = \frac{\rho w_0^2}{2\left(w_0^2\rho^2 N + \frac{1}{\sigma^2}\right)} \tag{41}$$

The larger the number of parameters N, the smaller the optimal learning step γ^*_{N1}. Adding parameters, increases the slope and the curvature in the descent direction hence a smaller step is needed to descend by the same amount. Hence, by adding parameters, learning can reach the same speed for a smaller learning step.

Furthermore, if the signal to noise ratio σ decreases, the direction of descent is less correlated with the gradient direction, hence a larger learning step can be tolerated. For this learning step, the optimal learning speed is

$$\mathbb{E}[\nu_t]^*_{N1} = \frac{1}{4}\frac{(\rho^2 w_0^2 N)^2}{w_0^2\rho^3 N^2 + \frac{\rho}{\sigma^2}N} \tag{42}$$

Note that the optimal learning speed saturates as N increases reaching the asymptote

$$\lim_{N\to\infty} \mathbb{E}[\nu_t]^*_{N1} = \frac{\rho w_0}{4} \tag{43}$$

Figure **??** shows the learning speed as a function of learning step for different sized networks. As expected, the optimal learning step decreases for network size. Figure **??** B) shows the optimal learning speed as a function of N for different ρ. The larger the average curvature ρ, the faster the optimal learning speed saturates. Hence, adding parameters is particularly beneficial when the average curvature is small.

However, learning performance is not only about maximising learning speed, it is also about minimising steady state loss. We can compute the local task difficulty given the weight state $\mathbf{w}_t = \frac{w_{SS}}{\sqrt{N}}(1,1,\cdots,1)^T$, where w_{SS} determines the distance of the weight point from the minimum. [it makes more sense to take a poitn $\mathbf{w}_t = w_{SS}(1,1,\cdots,1)$] As mentioned for the 1D case, the local task difficulty is only relevant near the minimum of the function. Thus we will have

$|w_{SS}|$ close to zero (as opposed to w_0 which is far from zero).

$$\mathbb{E}[G_t]_{N1} = \gamma \frac{1}{||\nabla f_{N1}(\mathbf{w}_t)||^2} \nabla f_{N1}(\mathbf{w}_t)^T H_{N1} \nabla f_{N1}(\mathbf{w}_t) + \frac{\eta^2}{\gamma} \frac{1}{||\nabla f_{N1}(\mathbf{w}_t)||^2} \mathbb{E}[\epsilon_t^T H_{N1} \epsilon_t] \tag{44}$$

$$= \gamma \rho N + \frac{\eta^2}{\gamma} \frac{1}{\rho N (w_{1t})^2} \tag{45}$$

$$= \gamma \rho N + \frac{\eta^2}{\gamma} \frac{1}{\rho w_{SS}^2} \tag{46}$$

the first term in the local task difficulty increases as the number of dimensions N increases and the second term is constant. For fixed σ

$$\mathbb{E}[G_t]_{N1} = \gamma \rho N + \frac{\gamma}{\sigma^2} \frac{1}{\rho w_{SS}^2} \tag{47}$$

The local task difficulty increase linearly with γ and N. At first glance, it seems that adding parameters is detrimental for steady state performance. However, if we consider both learning speed and steady state loss simultaneously, this is not the full story.

In machine learning algorithms, the learning step γ is often decreased during learning to achieve both large learning speed and small steady state loss. This requires knowledge of the value of the loss function or its gradient during learning to modulate the learning step. However, in biological learning, it is often hard to access the exact value of the loss or even modify the learning step during learning without affecting other learning processes. It is therefore important to consider the case where γ is constant throughout learning.

In that case, there is a trade-off between learning speed and steady state loss. Increasing γ up to γ_{N1}^* improves the learning speed but worsens the steady state loss. We can compute the local task difficulty for optimal learning speed $\gamma = \gamma_{N1}^*$

$$\mathbb{E}[G_t]_{N1}^* = \left(\frac{\rho w_0^2}{2\left(w_0^2 \rho^2 N + \frac{1}{\sigma^2}\right)} \right) \left(\rho N + \frac{1}{\sigma^2} \frac{1}{\rho w_{SS}^2} \right) \tag{48}$$

$$= \frac{1}{2} \left(\frac{\sigma^2 \rho w_0^2}{\sigma^2 w_0^2 \rho^2 N + 1} \right) \left(\frac{\sigma^2 \rho^2 w_{SS}^2 N + 1}{\sigma^2 \rho w_{SS}^2} \right) \tag{49}$$

$$= \frac{1}{2} \frac{w_0^2}{w_{SS}^2} \frac{\sigma^2 \rho^2 w_{SS}^2 N + 1}{\sigma^2 w_0^2 \rho^2 N + 1} \tag{50}$$

$$\tag{51}$$

It is not obvious how this quantity changes with N from this formula alone. We

can take the derivative with respect to N

$$\frac{d}{dN}\mathbb{E}[G_t]^*_{N1} = \frac{1}{2}\frac{w_0^2}{w_{SS}^2}\frac{\left(\sigma^2\rho^2 w_{SS}^2\right)\left(\sigma^2 w_0^2\rho^2 N+1\right)-\sigma^2 w_0^2\rho^2\left(\sigma^2\rho^2 w_{SS}^2 N+1\right)}{(\sigma^2 w_0^2\rho^2 N+1)^2} \tag{52}$$

$$= \frac{1}{2}\frac{w_0^2}{w_{SS}^2}\frac{\left(\sigma^2\rho^2\right)\left(\sigma^2\rho^2 w_{SS}^2 w_0^2 N+w_{SS}^2-\rho^2\sigma^2 w_{SS}^2 w_0^2 N-w_0^2\right)}{(\sigma^2 w_0^2\rho^2 N+1)^2} \tag{53}$$

$$= -\frac{1}{2}\frac{w_0^2}{w_{SS}^2}\frac{\left(\sigma^2\rho^2\right)\left(w_0^2-w_{SS}^2\right)}{(\sigma^2 w_0^2\rho^2 N+1)^2} \tag{54}$$

$$\tag{55}$$

As $w_{ss}^2 << w_0^2$, this derivative is negative for all N. So, the local task difficulty for $\gamma = \gamma^*_{N1}$ decreases with the network size. Adding parameters to the quadratic function with this type of expansion allows for better learning speed and steady state loss simultaneously. This expansion adds zero eigenvalues to the hessian of the function while maintaining the average curvature. Hence, the direction of non-zero curvature have larger slope and curvature.

We have shown that this particular change in loss landscape is beneficial for learning with "noisy gradient descent". The learning speed, which is strongly dependent on the slope, increases with the expansion. The steady state loss, which is strongly dependent on the ratio between the average eigenvalue and the slope, is also improved. As N increases, we can always find a learning step γ that improves learning performance.

All equal eigenvalues The second type of functions are $f_{N2} : \mathbb{R}^N \to \mathbb{R}$ with

$$f_{N2}(\mathbf{w}) = \frac{\rho}{2}||\mathbf{w}||^2 = \frac{\rho}{2}(w_1^2 + w_2^2 + \cdots + w_N^2) \tag{56}$$

where $\mathbf{w} = (w_1, w_2, \cdots, w_N)^T \in \mathbb{R}^N$. The gradient of this function is the vector

$$\nabla f_{N2}(\mathbf{w}) = \rho \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix} \tag{57}$$

meaning that the slope is proportional to w_i in every direction. The hessian matrix is constant in $\mathbb{R}$ and given by

$$H_{N2} \equiv \nabla^2 f_{N2}(\mathbf{w}) = \begin{pmatrix} \rho & 0 & \cdots & 0 \\ 0 & \rho & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & & \rho \end{pmatrix} \tag{58}$$

This matrix has N orthogonal eigenvectors $e_i, \quad i = 1, \cdots, N$ all the eigenvalues equal to $\lambda_i = \rho$. As expected, the average curvature given by the average eigenvalue is

$$\frac{Tr(H_{N2})}{N} = \rho \tag{59}$$

The spread of the eigenvalues

$$\frac{Tr((H_{N2})^3)}{Tr\left((H_{N2})^2\right)} = \rho \tag{60}$$

is equal to the average eigenvalue. This type of expansion is qualitatively different. In this case, the average eigenvalue and spread of eigenvalues are maintained for all N. Furthermore, the gradient norm $||\nabla f_{N2}(\mathbf{w})||$ is also constant if the distance to the origin $||\mathbf{w}||$ is maintained with the expansion.

As in the previous case, we can compute the expected learning speed and local task difficulty at $\mathbf{w}_t = \frac{w_0}{\sqrt{N}}(1, 1, \cdots, 1)^T$. We can compute the learning speed

$$\mathbb{E}[\nu_t]_{N2} = \gamma||\nabla f_{N2}(\mathbf{w}_t)||^2 - \gamma^2 \nabla f_{N2}(\mathbf{w}_t)^T H_{N2} \nabla f_{N2}(\mathbf{w}_t) - \eta^2 \mathbb{E}[\epsilon_t^T H_{N2} \epsilon_t] \tag{61}$$

$$= \gamma\rho^2||\mathbf{w}_t||^2 - \gamma^2\rho^3||\mathbf{w}_t||^2 - \eta^2 \frac{Tr(H_{N1})}{N}\mathbb{E}[||\epsilon_t||^2] \tag{62}$$

$$= \gamma\rho^2||\mathbf{w}_t||^2 - \gamma^2\rho^3||\mathbf{w}_t||^2 - \eta^2\rho\mathbb{E}[||\epsilon_t||^2] \tag{63}$$

$$= \gamma\rho^2 w_0^2 - \gamma^2\rho^3 w_0^2 - \eta^2\rho N \tag{64}$$

where we have used the face that each element of the vector ϵ_t is drawn from a random normal distribution $\mathcal{N}(0, 1)$ so the expected $\mathbb{E}[||\epsilon_t||^2] = N\mathbb{E}[\epsilon_t^2] = N$. For fixed σ, we get

$$\mathbb{E}[\nu_t]_{N2} = \gamma\rho^2 w_0^2 - \gamma^2\rho^3 w_0^2 - \left(\frac{\gamma}{\sigma}\right)^2 \rho N \tag{65}$$

The learning speed $\mathbb{E}[\nu_t]_{N2}$ decreases linearly with N. Geometrically, this is due to the fact that the slope and curvature terms are independent of N while the norm of the noise increases with N. Hence, this expansion is not beneficial for learning speed improvement.

Furthermore, the learning speed is a quadratic function in the learning step γ; it increases with γ until it reaches a maximal value for γ^*_{N2}.

$$\gamma^*_{N2} = \frac{\rho w_0^2}{2\left(w_0^2\rho^2 + \frac{1}{\sigma^2}N\right)} \tag{66}$$

The larger the number of parameters N, the smaller the optimal learning step γ^*_{N2}. However, γ^*_{N2} decreases slowly with N. Indeed, far from the minimum (i.e. $|w_0| >> 0$), we have $\frac{N}{\sigma^2} << w_0^2\rho^2$, hence γ^*_{N2} is dominated by $\frac{\rho w_0^2}{2\rho^2 w_0^2}$.

Geometrically, the learning step is mainly determined by the slope which is constant with N.

The optimal learning speed for $\gamma = \gamma^*_{N2}$ is

$$\mathbb{E}[\nu_t]^*_{N2} = \frac{1}{4}\frac{\rho^3 w_0^4}{w_0^2\rho^2 + \frac{N}{\sigma^2}} \tag{67}$$

The optimal learning speed decreases slowly as we increase N.

Similarly, we can compute the local task difficulty given the weight state $\mathbf{w}_t = \frac{w_{SS}}{\sqrt{N}}(1, 1, \cdots, 1)^T$, where w_{SS} determines the distance of the weight point from the minimum. [it makes more sense to take a poitn $\mathbf{w}_t = w_{SS}(1, 1, \cdots, 1)$]

$$\mathbb{E}[G_t]_{N2} = \gamma\frac{1}{||\nabla f_{N2}(\mathbf{w}_t)||^2}\nabla f_{N2}(\mathbf{w}_t)^T H_{N2}\nabla f_{N2}(\mathbf{w}_t) + \frac{\eta^2}{\gamma}\frac{1}{||\nabla f_{N2}(\mathbf{w}_t)||^2}\mathbb{E}[\epsilon_t^T H_{N2}\epsilon_t] \tag{68}$$

$$= \gamma\rho + \frac{\eta^2}{\gamma}\frac{N\rho}{\rho^2||\mathbf{w}_t||^2} \tag{69}$$

$$= \gamma\rho + \frac{\eta^2}{\gamma}\frac{N}{\rho w_{SS}^2} \tag{70}$$

the first term in the local task difficulty is constant with N and the second increases. For fixed σ

$$\mathbb{E}[G_t]_{N2} = \gamma\rho + \frac{\gamma}{\sigma^2}\frac{N}{\rho w_{SS}^2} \tag{71}$$

The local task difficulty increase linearly with γ and N. In this case, adding parameters is detrimental for steady state performance and learning speed both.

We can compute the local task difficulty for optimal learning speed $\gamma = \gamma^*_{N2}$

$$\mathbb{E}[G_t]^*_{N2} = \left(\frac{\rho w_0^2}{2\left(w_0^2\rho^2 + \frac{1}{\sigma^2}N\right)}\right)\left(\rho + \frac{1}{\sigma^2}\frac{N}{\rho w_{SS}^2}\right) \tag{72}$$

$$= \frac{1}{2}\frac{w_0^2}{w_{SS}^2}\frac{\rho^2 w_{ss}^2\sigma^2 + N}{w_0^2\rho^2\sigma^2 + N} \tag{73}$$

It is not obvious how this quantity changes with N from this formula alone. We can take the derivative with respect to N

$$\frac{d}{dN}\mathbb{E}[G_t]^*_{N2} = \frac{1}{2}\frac{w_0^2}{w_{SS}^2}\frac{\rho^2\sigma^2(w_0^2 - w_{SS}^2)}{(w_0^2\rho^2\sigma^2 + N)^2} \tag{74}$$

As $w_{SS}^2 << w_0^2$, this derivative is positive for all N. So, the local task difficulty for $\gamma = \gamma^*_{N2}$ increases with the network size. Adding parameters to the quadratic function with with this type of expansion worsens both learning speed and steady state loss.

This expansion adds eigenvalues with value ρ to the hessian of the function which maintains the average curvature. The slope in all directions remains the same as we add parameters and so does the curvature. This change in loss landscape is detrimental for learning "noisy gradient descent". The learning speed, which is strongly dependent on the slope, deteriorates slightly due to the noise. The steady state loss worsens due to the increase of the noise norm $\mathbb{E}[||\epsilon||^2]$.