Understanding the role of invariances in training neural networks

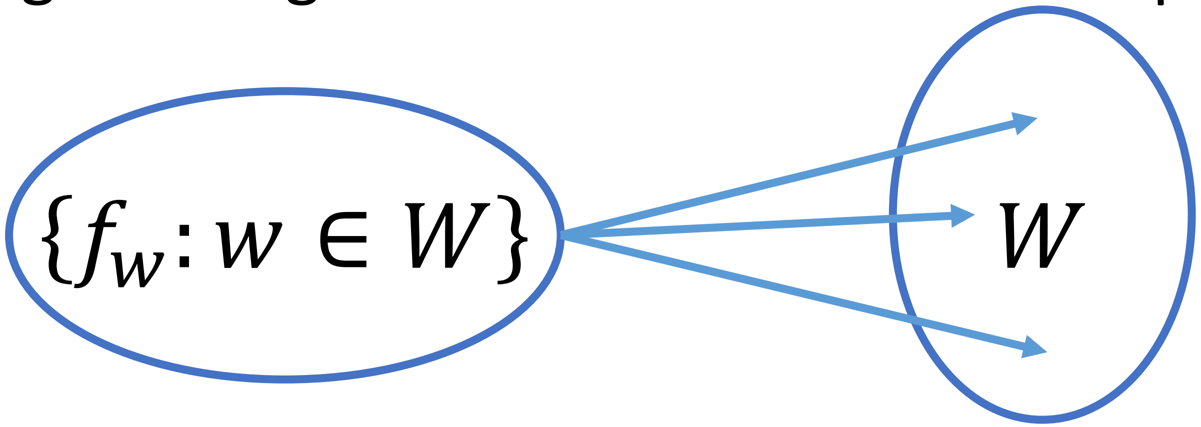
Ryota Tomioka

Microsoft Research Cambridge

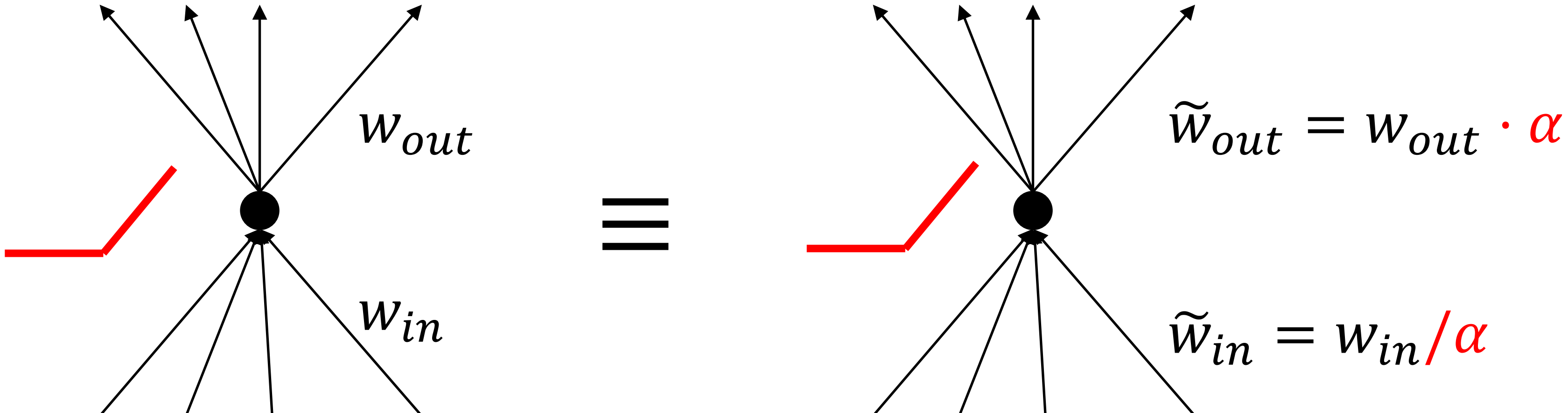
Joint work with:
Behnam Neyshabur, Ruslan Salakhutdinov, and Nathan Srebro

Neural networks are over-parametrized

- Many weight configurations realize the same input-output mapping



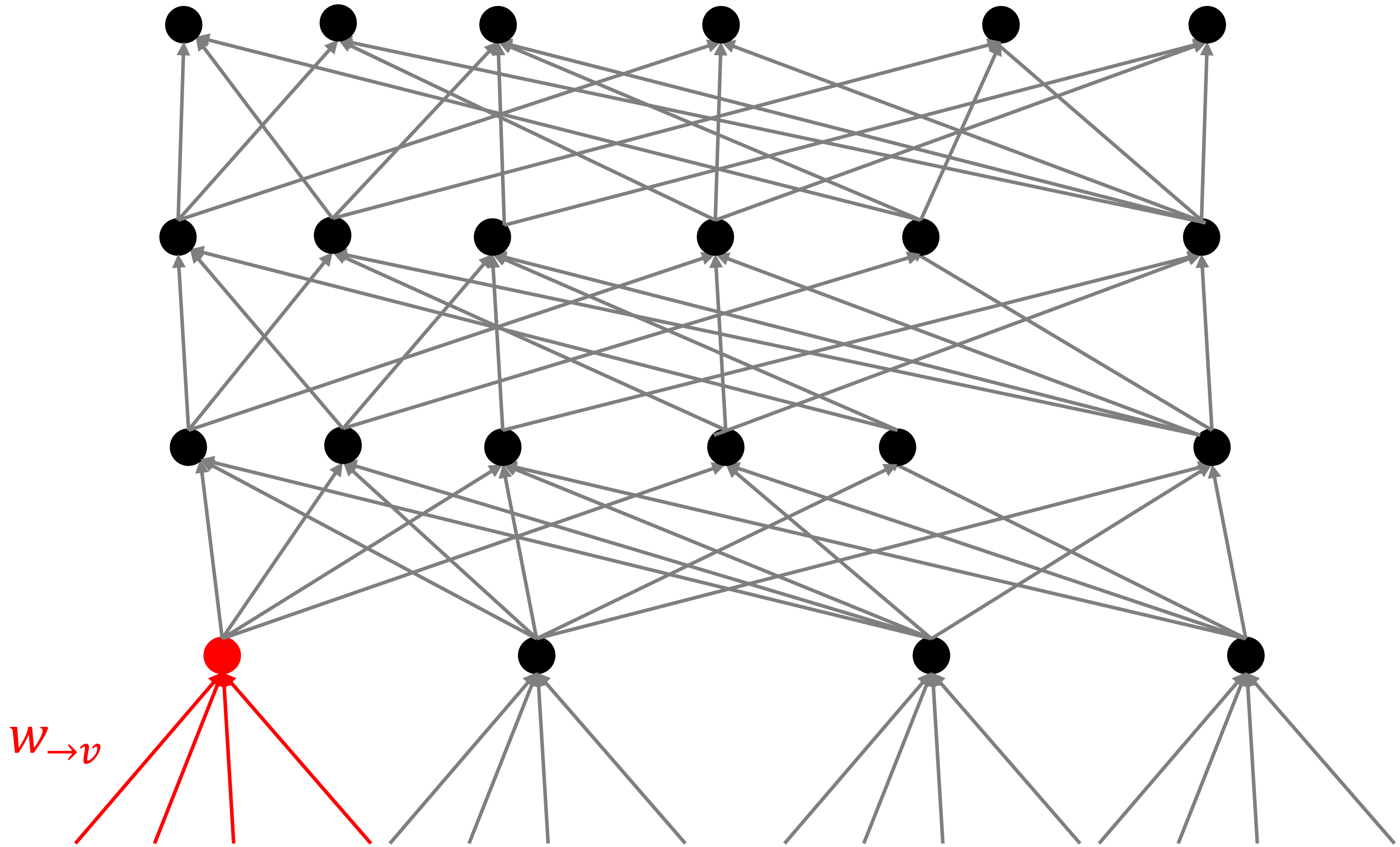
- Example: node-wise rescaling



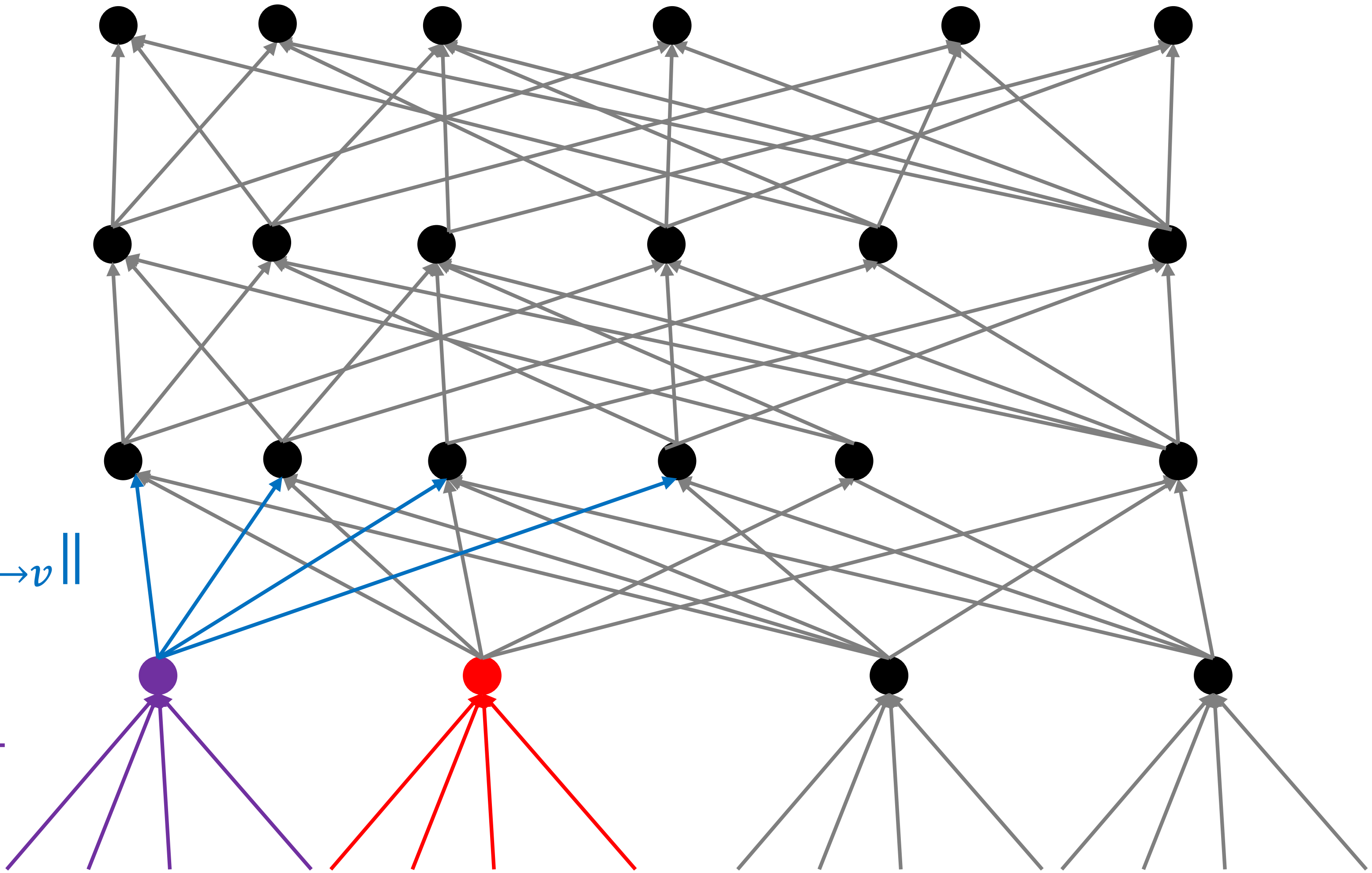
Questions

- What is the consequence?
  - Does it generalize better because it is over-parametrized?
  - Can we optimize better if we are aware of the ambiguities?
- Basic question
  - Can we characterize what sort of ambiguities there are?

Node-wise rescaling



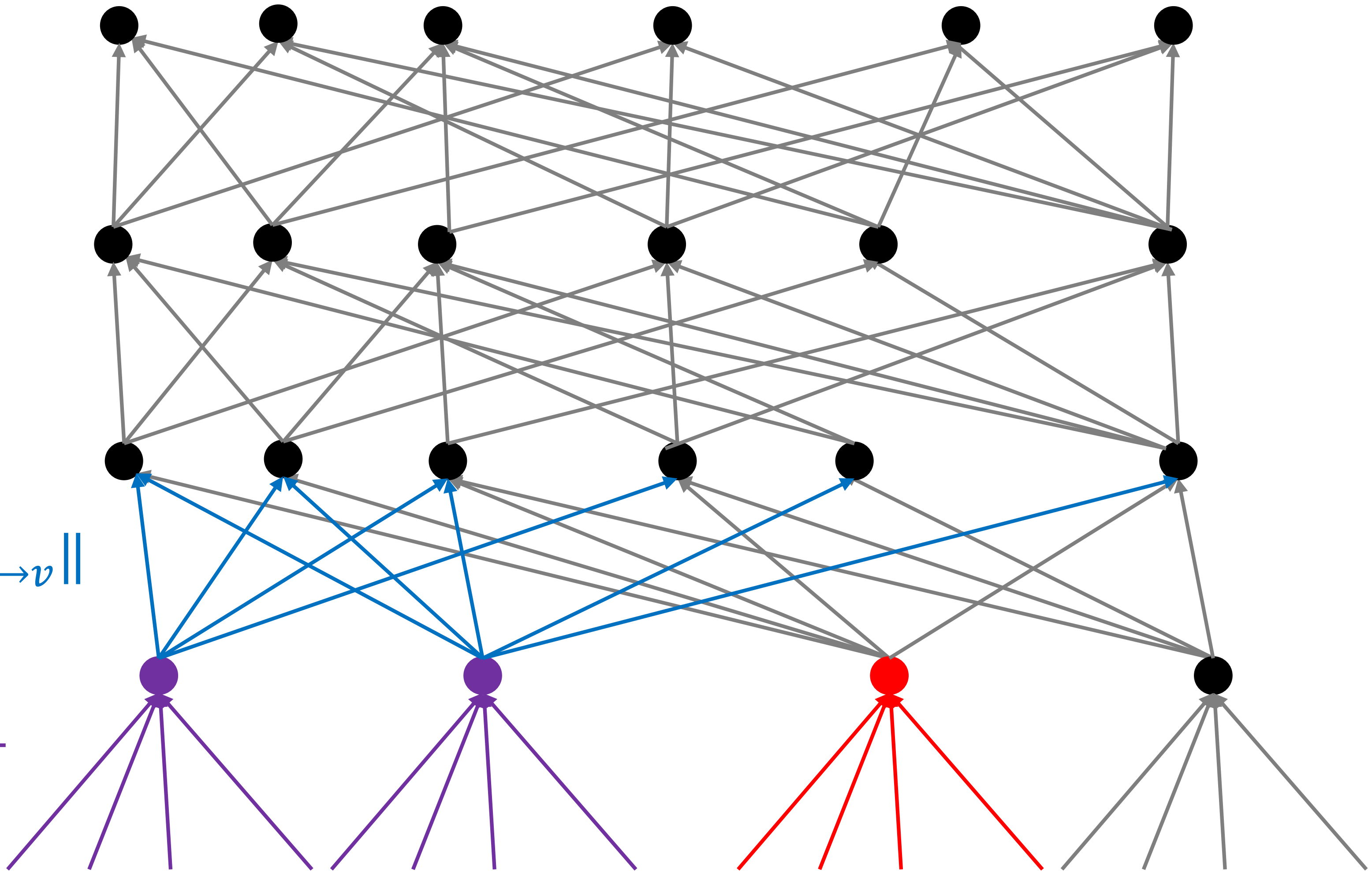
Node-wise rescaling



$$\tilde{w}_{v\to} = w_{v\to} \cdot \|w_{\to v}\|$$

$$\tilde{w}_{\to v} = \frac{w_{\to v}}{\|w_{\to v}\|}$$

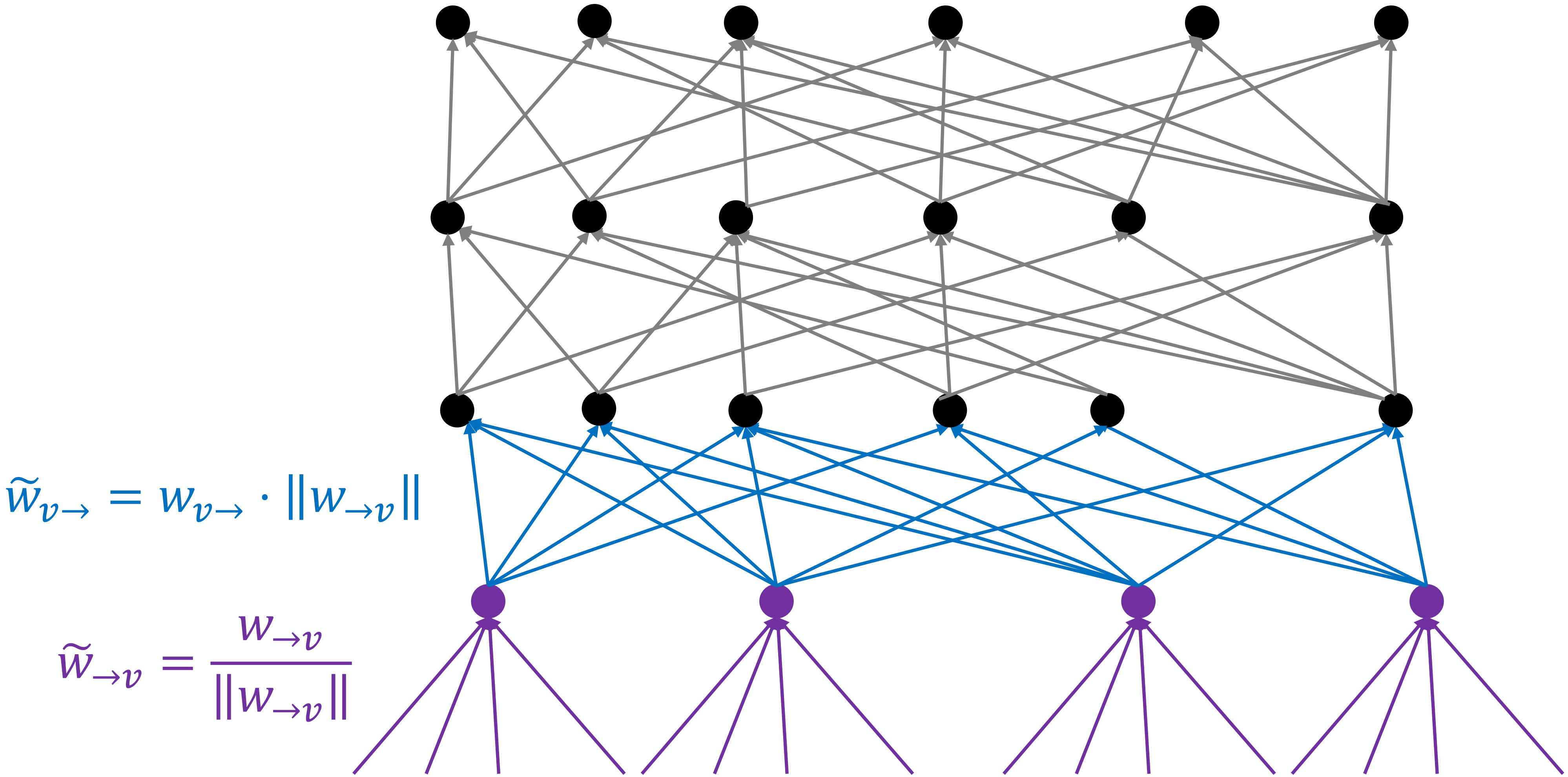
Node-wise rescaling



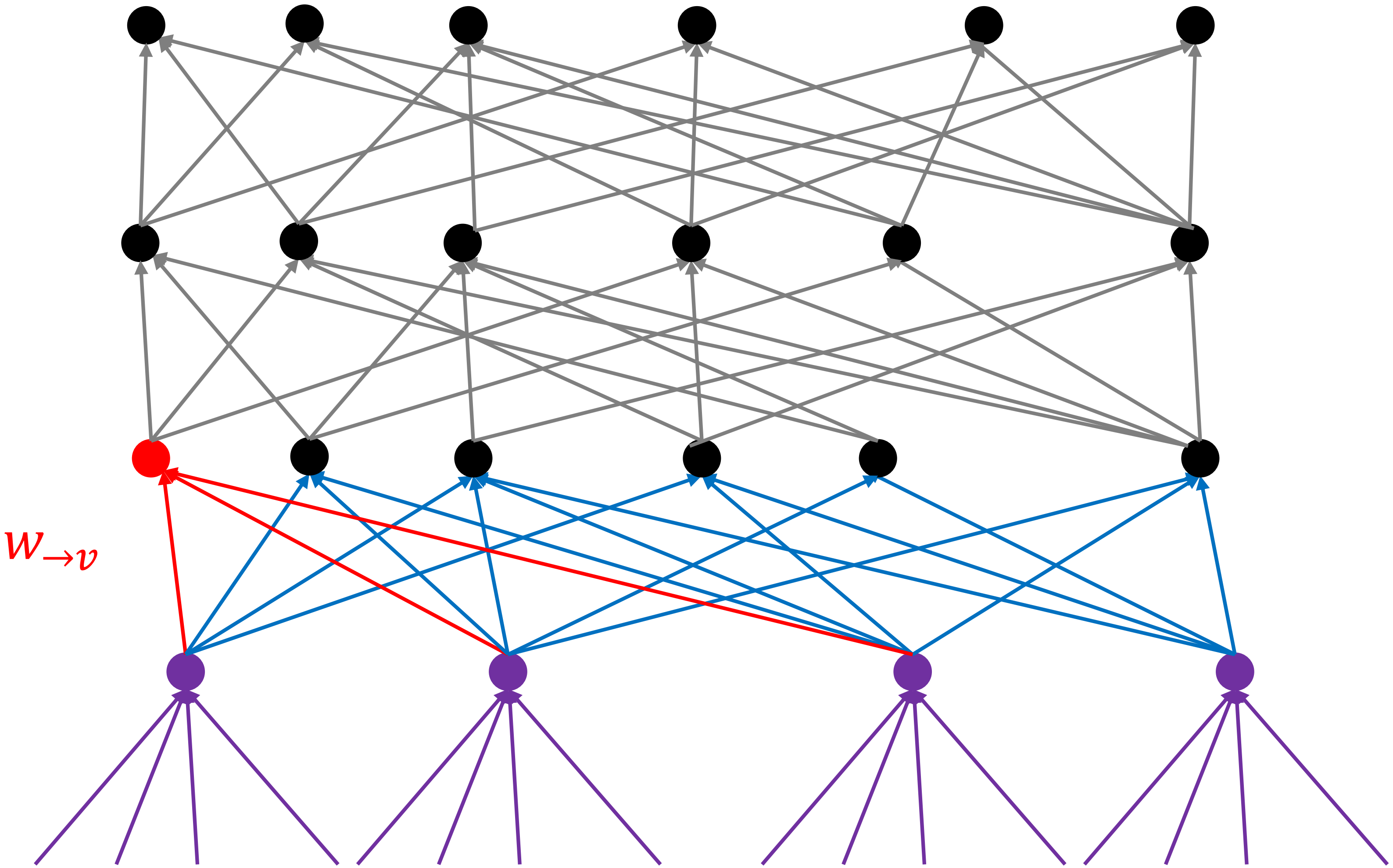
$$\tilde{w}_{v\rightarrow} = w_{v\rightarrow} \cdot \|w_{\rightarrow v}\|$$

$$\tilde{w}_{\rightarrow v} = \frac{w_{\rightarrow v}}{\|w_{\rightarrow v}\|}$$

Node-wise rescaling



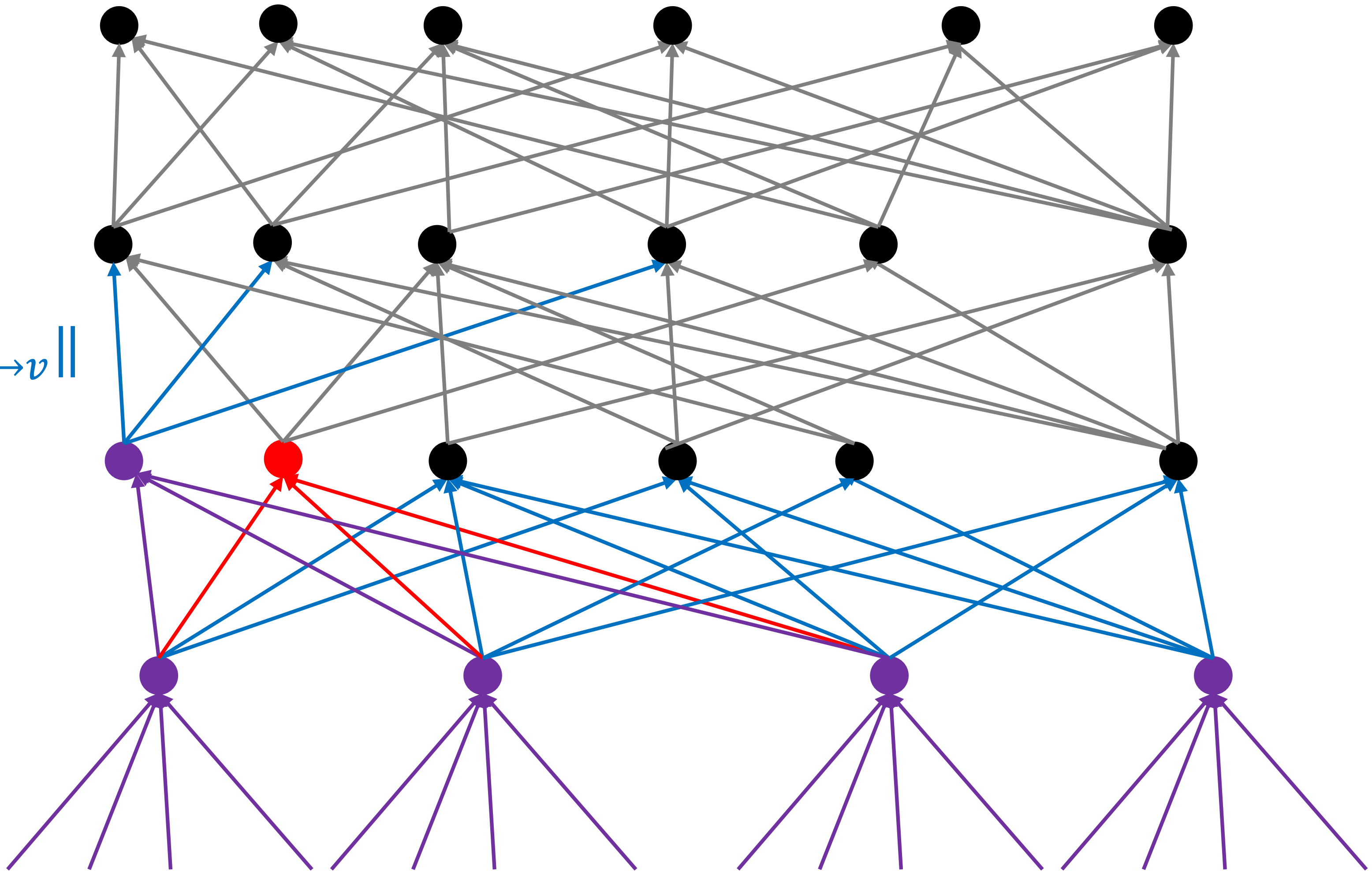
Node-wise rescaling



Node-wise rescaling

$$\widetilde{w}_{v\to} = w_{v\to} \cdot \|w_{\to v}\|$$

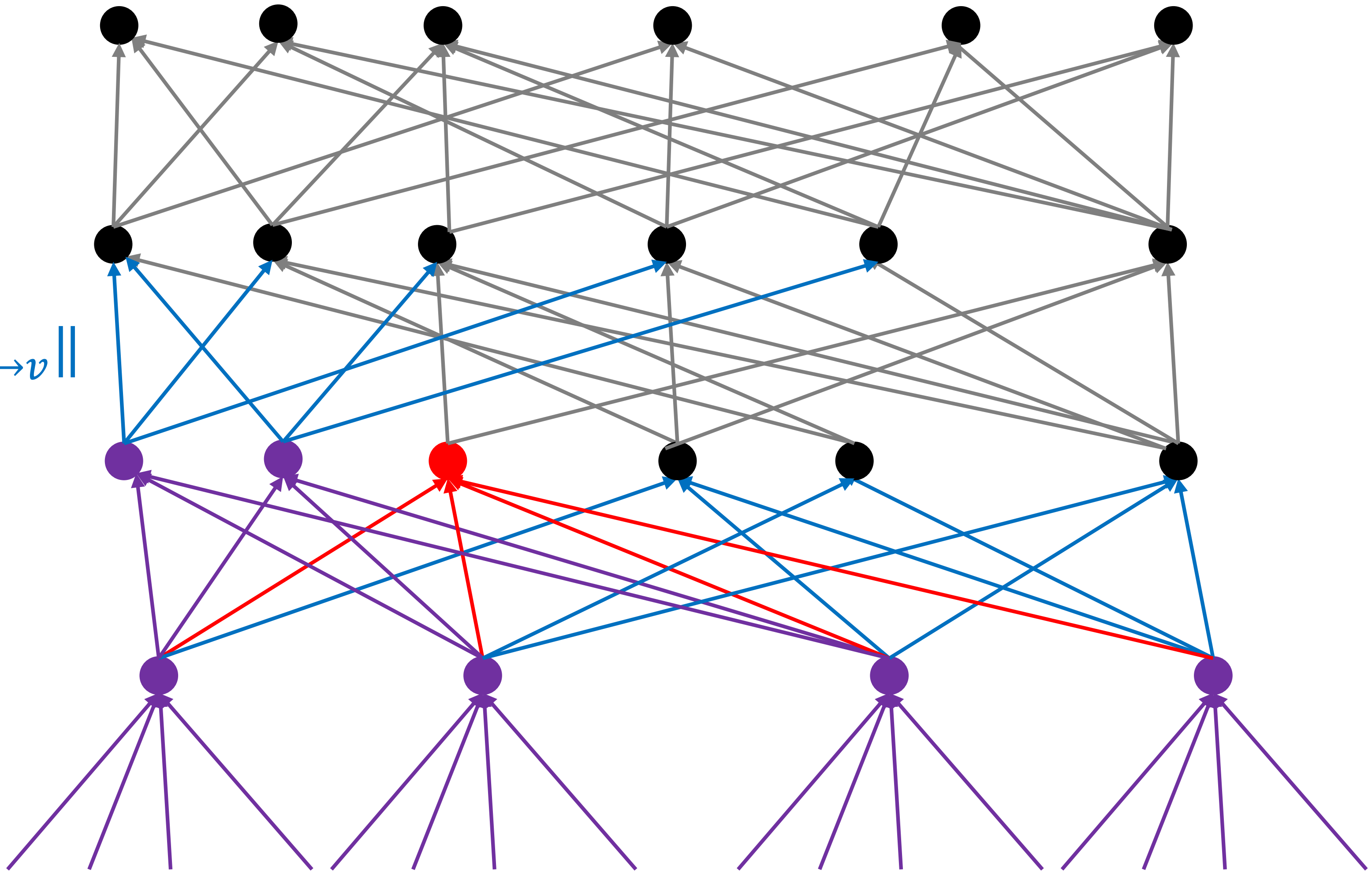
$$\widetilde{w}_{\to v} = \frac{w_{\to v}}{\|w_{\to v}\|}$$



Node-wise rescaling

$$\widetilde{w}_{v\to} = w_{v\to} \cdot \|w_{\to v}\|$$

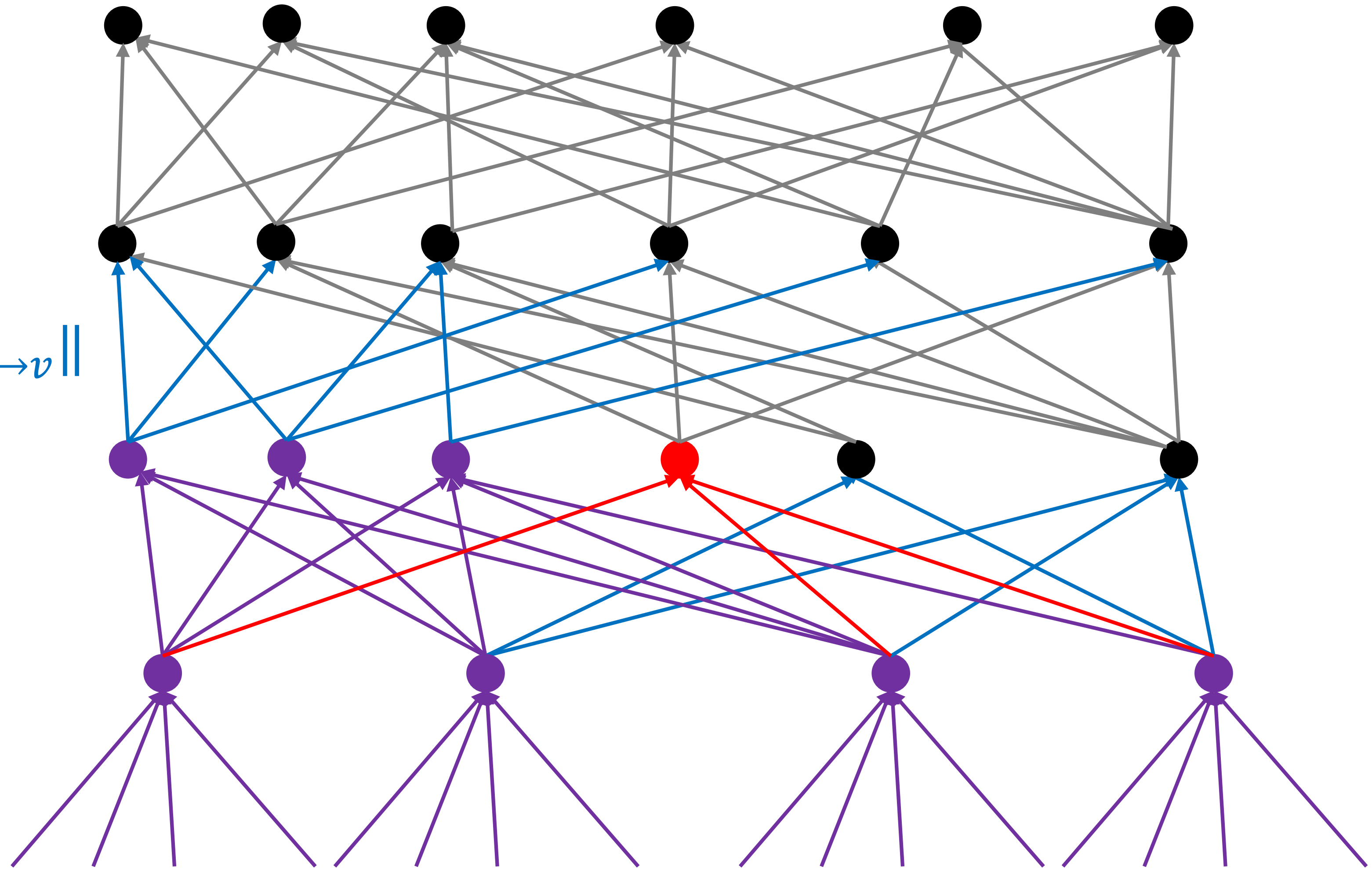
$$\widetilde{w}_{\to v} = \frac{w_{\to v}}{\|w_{\to v}\|}$$



Node-wise rescaling

$$\tilde{w}_{v\rightarrow} = w_{v\rightarrow} \cdot \|w_{\rightarrow v}\|$$

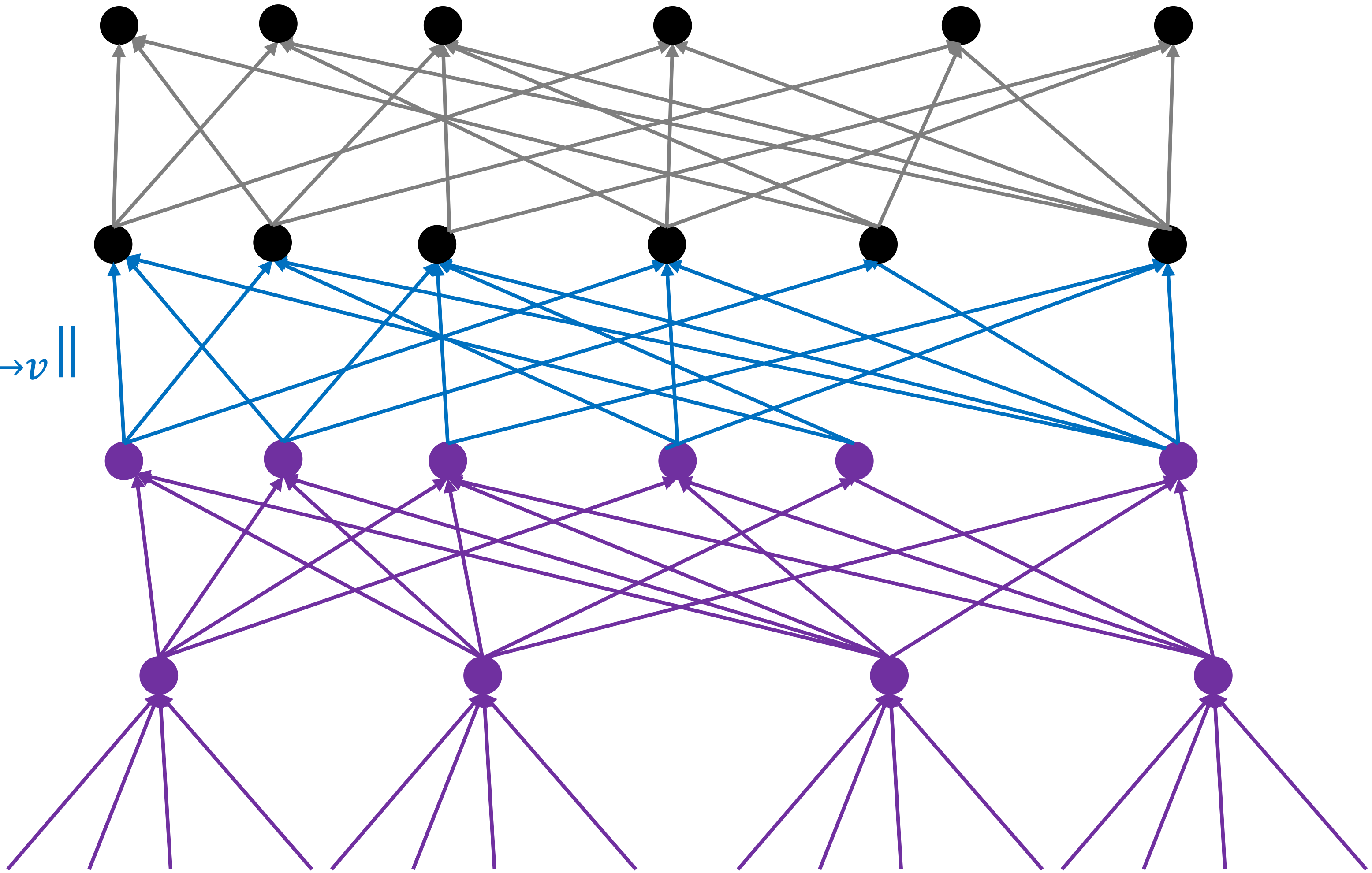
$$\tilde{w}_{\rightarrow v} = \frac{w_{\rightarrow v}}{\|w_{\rightarrow v}\|}$$



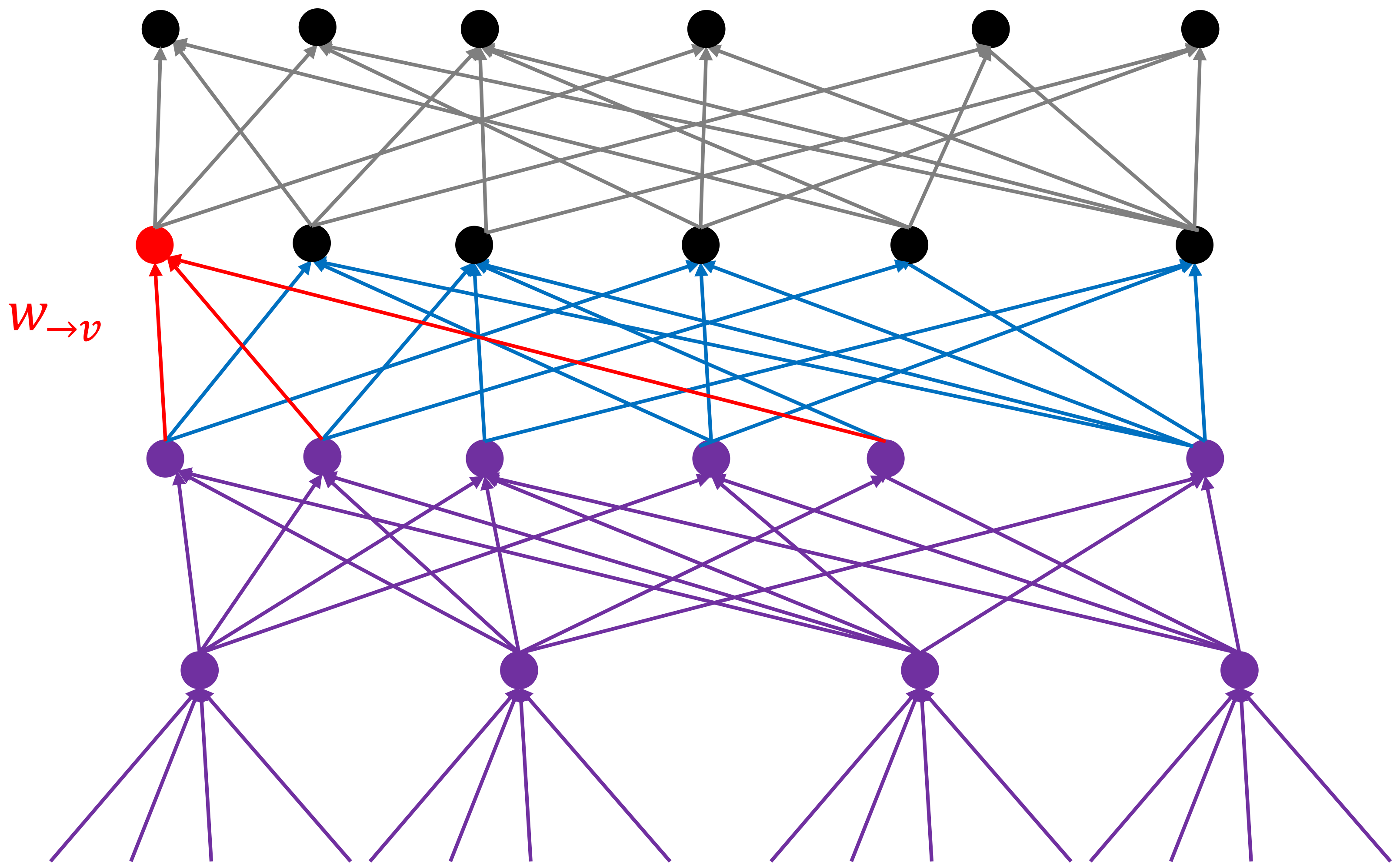
Node-wise rescaling

$$\tilde{w}_{v\rightarrow} = w_{v\rightarrow} \cdot \|w_{\rightarrow v}\|$$

$$\tilde{w}_{\rightarrow v} = \frac{w_{\rightarrow v}}{\|w_{\rightarrow v}\|}$$



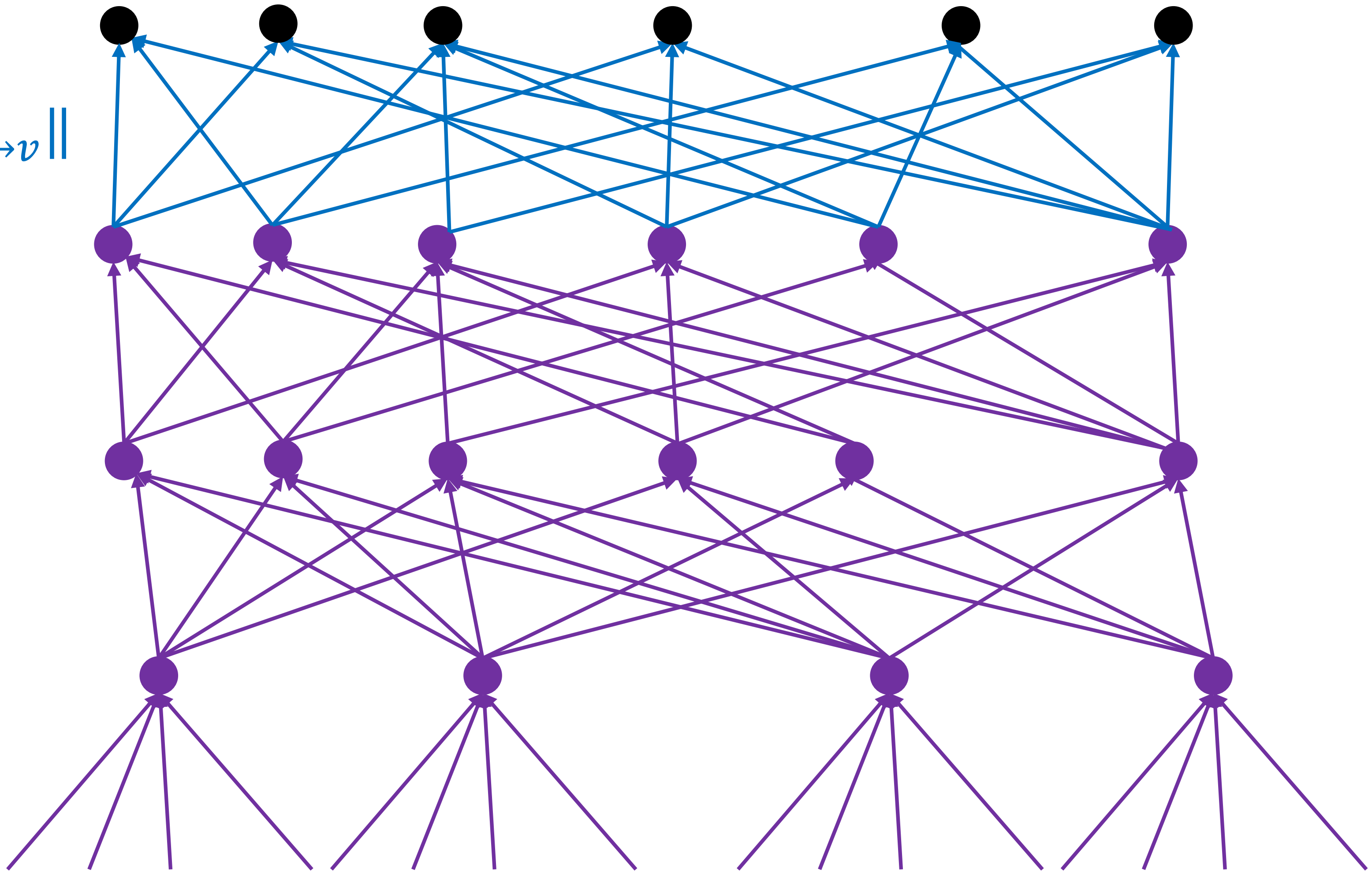
Node-wise rescaling



Node-wise rescaling

$$\tilde{w}_{v\rightarrow} = w_{v\rightarrow} \cdot \|w_{\rightarrow v}\|$$

$$\tilde{w}_{\rightarrow v} = \frac{w_{\rightarrow v}}{\|w_{\rightarrow v}\|}$$



What this means

- When the activation is rectified linear, the scale of the weights except for the last layer, carries no meaning
  - It doesn't change the function
  - In particular

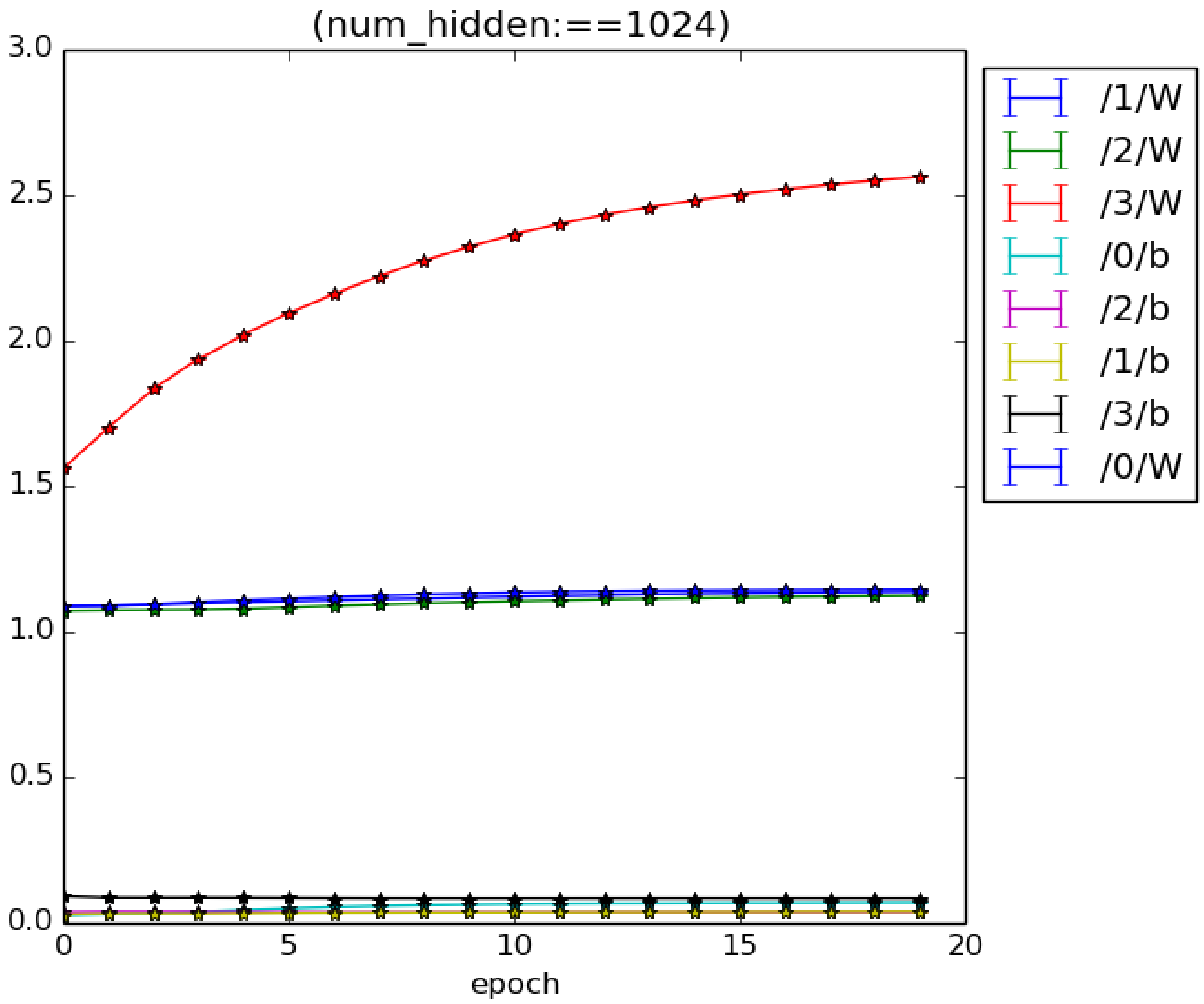
$$\underbrace{\{f_w\}}_{\text{Unconstrained parameterization}} \equiv \underbrace{\{f_w : \|w_{\to v}\|_2 = 1 \text{ if } v \notin V_{\text{out}}\}}_{\text{Constrained parameterization}} \equiv \underbrace{\left\{f_w : w_{\to v} = \frac{\widetilde{w}_{\to v}}{\|\widetilde{w}_{\to v}\|_2} \text{ if } v \notin V_{\text{out}}\right\}}_{\text{Normalized parameterization}}$$

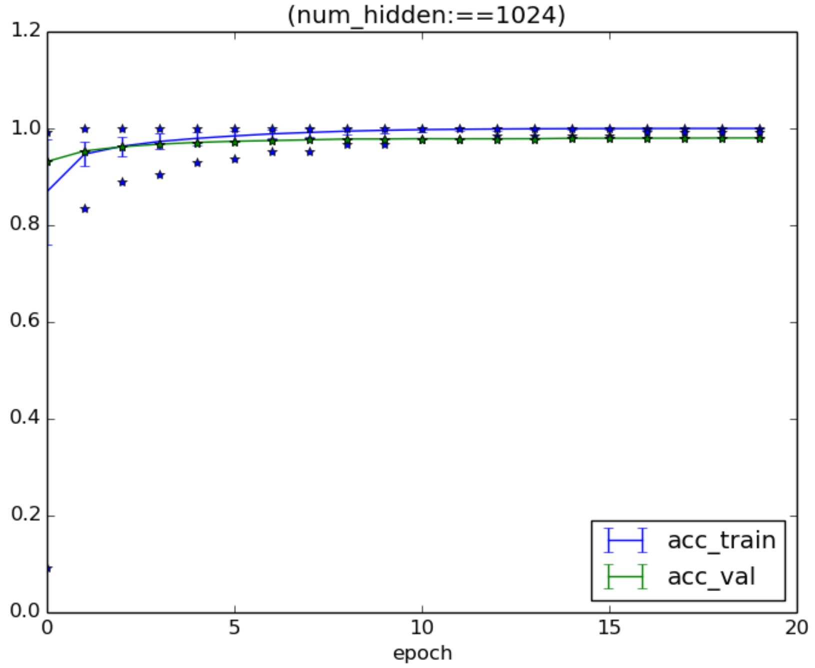
Unconstrained parameterization

Constrained parameterization

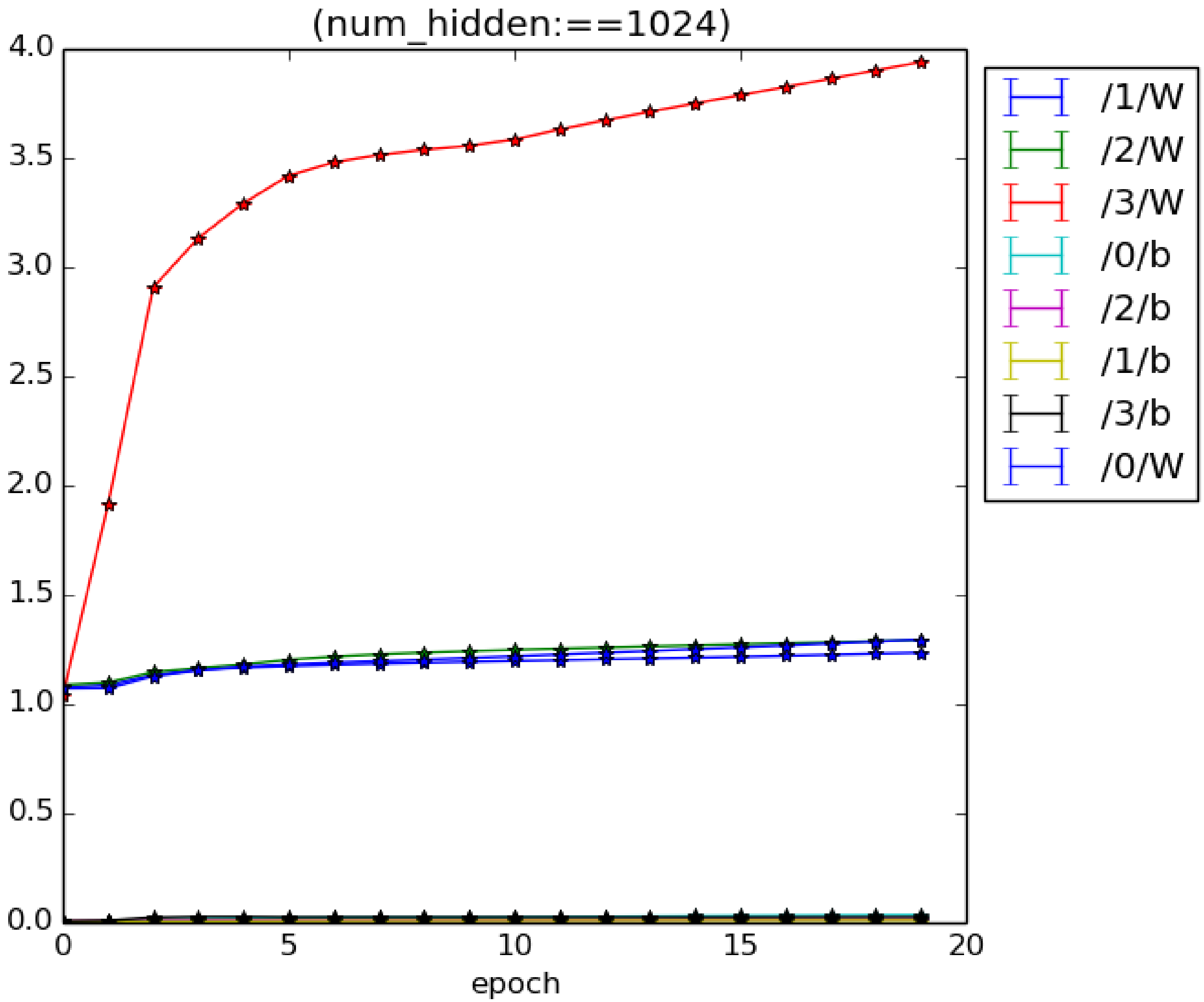
Normalized parameterization

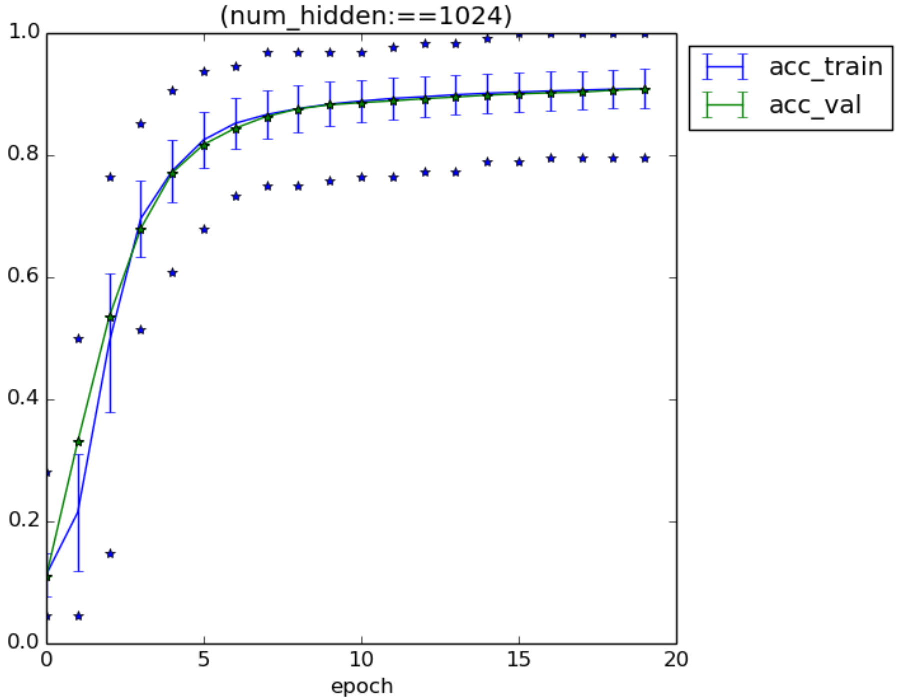
Check on MNIST (ReLU network)





Check on MNIST (sigmoid)





Theorem

- If the function $f_{w+\Delta w} = f_w + O(\|\Delta w\|^2)$, then

$$\left\langle \Delta w, \frac{\partial L(f_w)}{\partial w} \right\rangle = 0$$

  - That is, gradient is orthogonal to any direction that keeps the function unchanged

- Proof

$$\frac{\partial L(f_w)}{\partial w} = \frac{\partial L}{\partial f_w} \cdot \frac{\partial f_w}{\partial w} \quad \text{and} \quad \frac{\partial f_w}{\partial w} \cdot \Delta w = 0$$

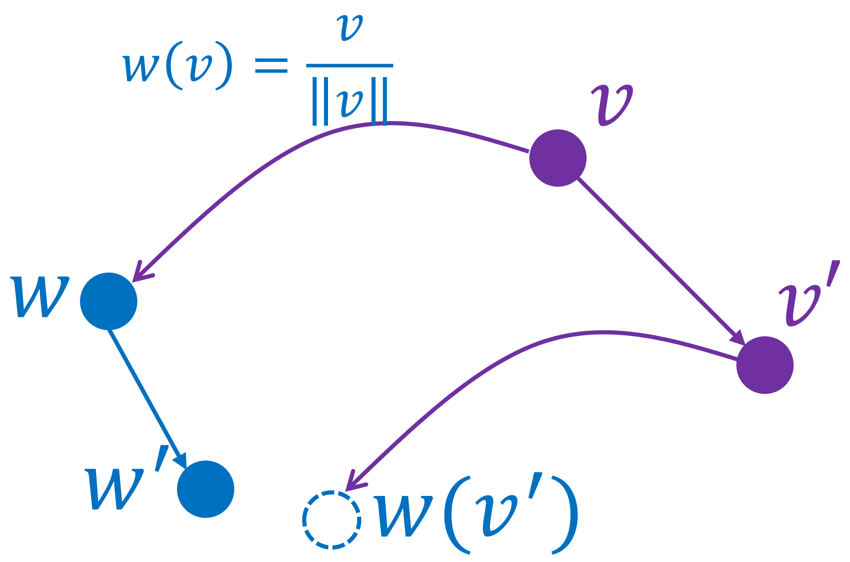
Is this good enough?

- Consider
  - Parametrization 1: $\{f_w\}$

$$w' = w - \eta \frac{\partial L}{\partial w}$$

  - Parametrization 2: $\left\{f_w : w = \frac{v}{\|v\|}\right\}$

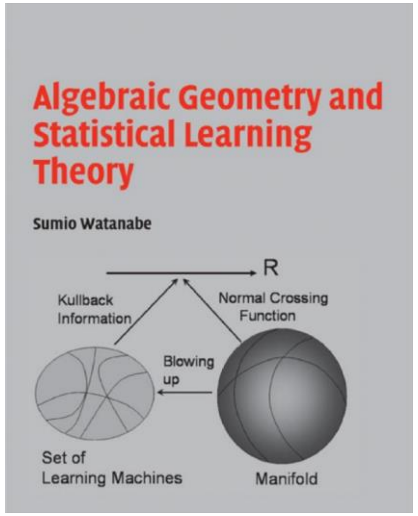
$$v' = v - \eta \frac{\partial L}{\partial w} \cdot \frac{\partial w}{\partial v}$$



Gradient descent in the two parametrizations are not equivalent!

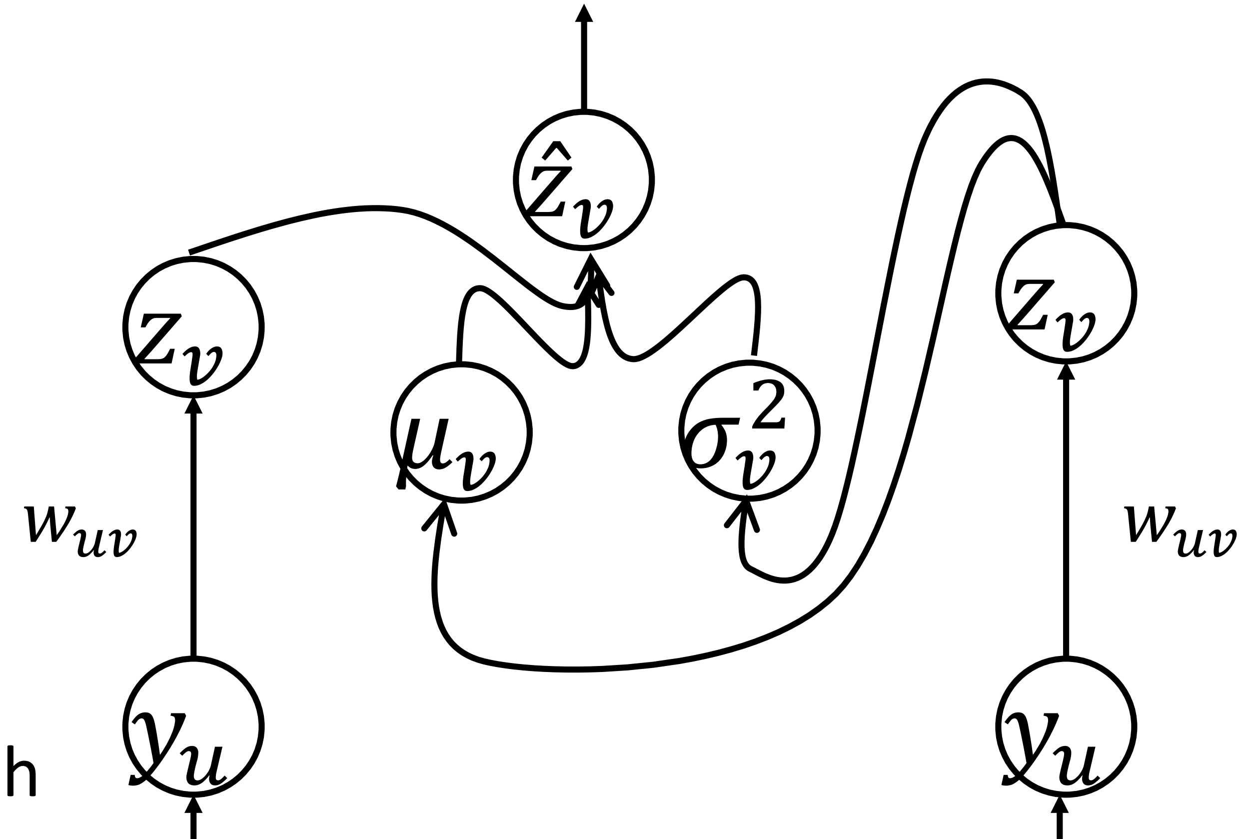
Two views

- Since many learning algorithms are parameterization sensitive, let's find a good parameterization.
  - Batch normalization [Ioffe &Szegedy, 2015]
  - Algebraic analysis of Bayesian neural networks [Watanabe et al.]
- Search for an algorithm that is invariant to parameterization
  - Natural gradient
  - Path-SGD [Neyshabur+ 2015]
  - Bayesian neural network with Jeffrey's prior



Batch normalization [Ioffe &Szegedy, 2015]

- Idea: normalize the input each unit receives to have zero mean and unit variance. Mean and variance estimated using a minibatch.



$$\hat{z}_v = \frac{z_v - \mu_v}{\sigma_v}$$

$$z_v = \sum_u w_{uv} y_u$$

Training mini-batch

Normalization mini-batch

Batch normalization as data-dependent reparametrization

- Forward path:

$$\hat{z}_v = \underbrace{\frac{w_{\to v}^T}{\sqrt{w_{\to v}^T C w_{\to v}}}}_{\tilde{w}} \cdot y_c$$

This is not parametrization invariant but works well in practice!!

where

$$\underbrace{y_c = \left(y_u - \frac{1}{n}\sum_{i=1}^{n} y_u^{(i)}\right)}_{\text{Centered activation}} \text{ and } \underbrace{C = \frac{1}{n}\sum_{i=1}^{n} y_c^{(i)} y_c^{(i)T}}_{\text{Covariance of previous layer}}$$

Path-SGD [Neyshabur+ 2015]

- (Approximate) steepest descent with respect to the squared path norm

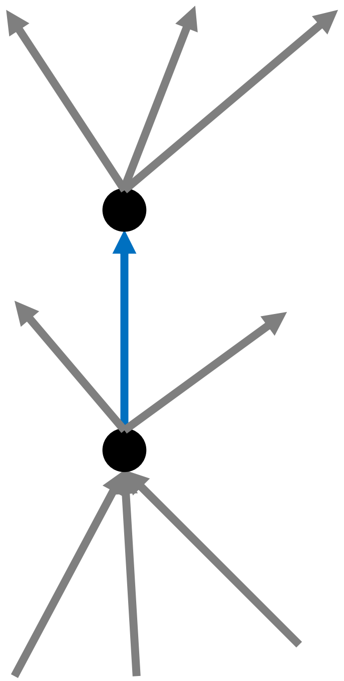
$$\Delta w_e = -\frac{1}{\kappa_e(w)} \cdot \frac{\partial L}{\partial w_e}$$

where

$$\kappa_e(w) = \sum_{p \ni e} \prod_{e' \in p \setminus \{e\}} w_{e'}^2$$

Sum over all the paths that include e

Product over all the edges along path p except for e



$\kappa_e(w)$ can be efficiently computed by forward and backward propagations.

Path-SGD [Neyshabur+ 2015]

- (Approximate) steepest descent with respect to the squared path norm

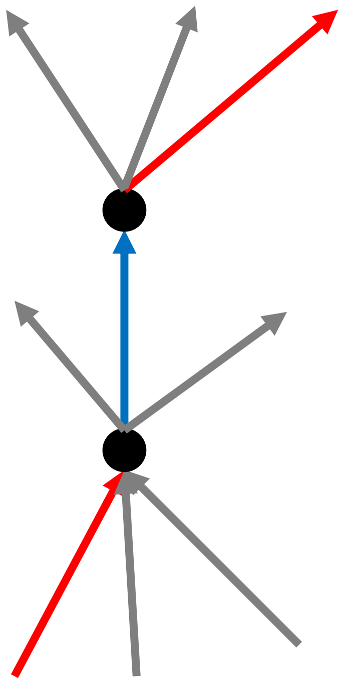
$$\Delta w_e = -\frac{1}{\kappa_e(w)} \cdot \frac{\partial L}{\partial w_e}$$

where

$$\kappa_e(w) = \sum_{p \ni e} \prod_{e' \in p \setminus \{e\}} w_{e'}^2$$

Sum over all the paths that include e

Product over all the edges along path p except for e



$\kappa_e(w)$ can be efficiently computed by forward and backward propagations.

Path-SGD [Neyshabur+ 2015]

- (Approximate) steepest descent with respect to the squared path norm

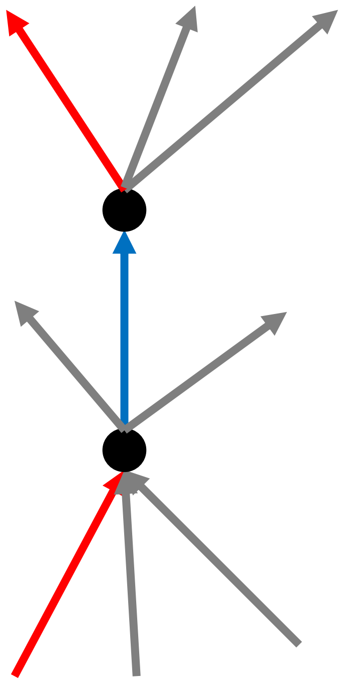
$$\Delta w_e = -\frac{1}{\kappa_e(w)} \cdot \frac{\partial L}{\partial w_e}$$

where

$$\kappa_e(w) = \sum_{p \ni e} \prod_{e' \in p \setminus \{e\}} w_{e'}^2$$

Sum over all the paths that include e

Product over all the edges along path p except for e



$\kappa_e(w)$ can be efficiently computed by forward and backward propagations.

Path-SGD [Neyshabur+ 2015]

- (Approximate) steepest descent with respect to the squared path norm

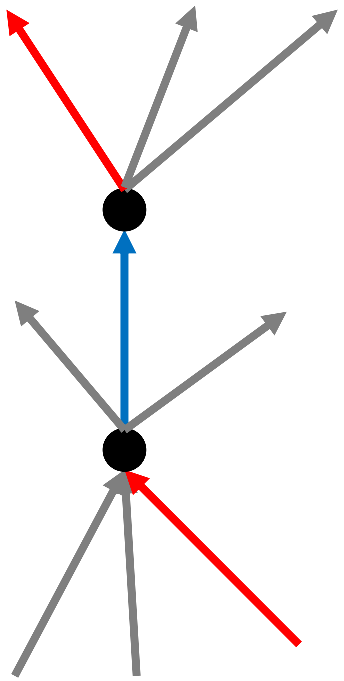
$$\Delta w_e = -\frac{1}{\kappa_e(w)} \cdot \frac{\partial L}{\partial w_e}$$

where

$$\kappa_e(w) = \sum_{p \ni e} \prod_{e' \in p \setminus \{e\}} w_{e'}^2$$

Sum over all the paths that include e

Product over all the edges along path p except for e



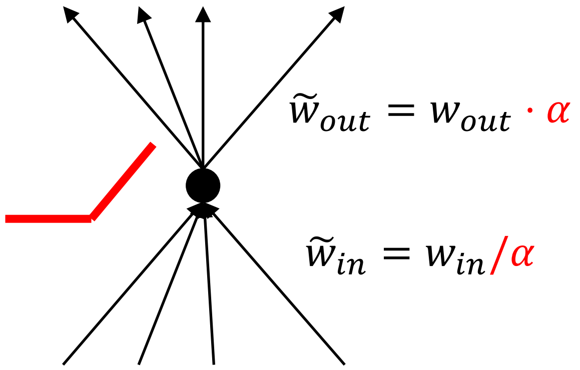
$\kappa_e(w)$ can be efficiently computed by forward and backward propagations.

Path-SGD is invariant to rescaling

- Suppose we transform

$$\tilde{w}_{v\rightarrow} = w_{v\rightarrow} \cdot \alpha, \qquad \tilde{w}_{\rightarrow v} = w_{\rightarrow v}/\alpha$$

Out-going edges to *v*   In-coming edges to *v*



- Then

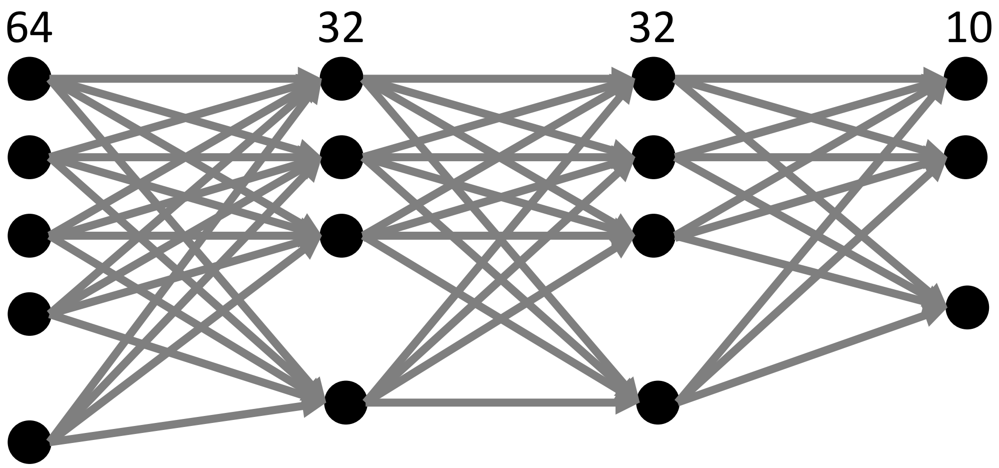
$$\Delta\tilde{w}_{v\rightarrow} = -\frac{\alpha^2}{\kappa_e(w)} \cdot \frac{1}{\alpha}\frac{\partial L}{\partial w_{v\rightarrow}} = \alpha \cdot \Delta w_{v\rightarrow}$$

$$\Delta\tilde{w}_{\rightarrow v} = -\frac{1}{\alpha^2 \cdot \kappa_e(w)} \cdot \alpha \cdot \frac{\partial L}{\partial w_{\rightarrow v}} = \alpha^{-1}\Delta w_{\rightarrow v}$$

This is invariant to node-wise rescaling (and works well)

Is node-wise rescaling all we should worry about?

- Network: 3 layers (64-32-32-10)



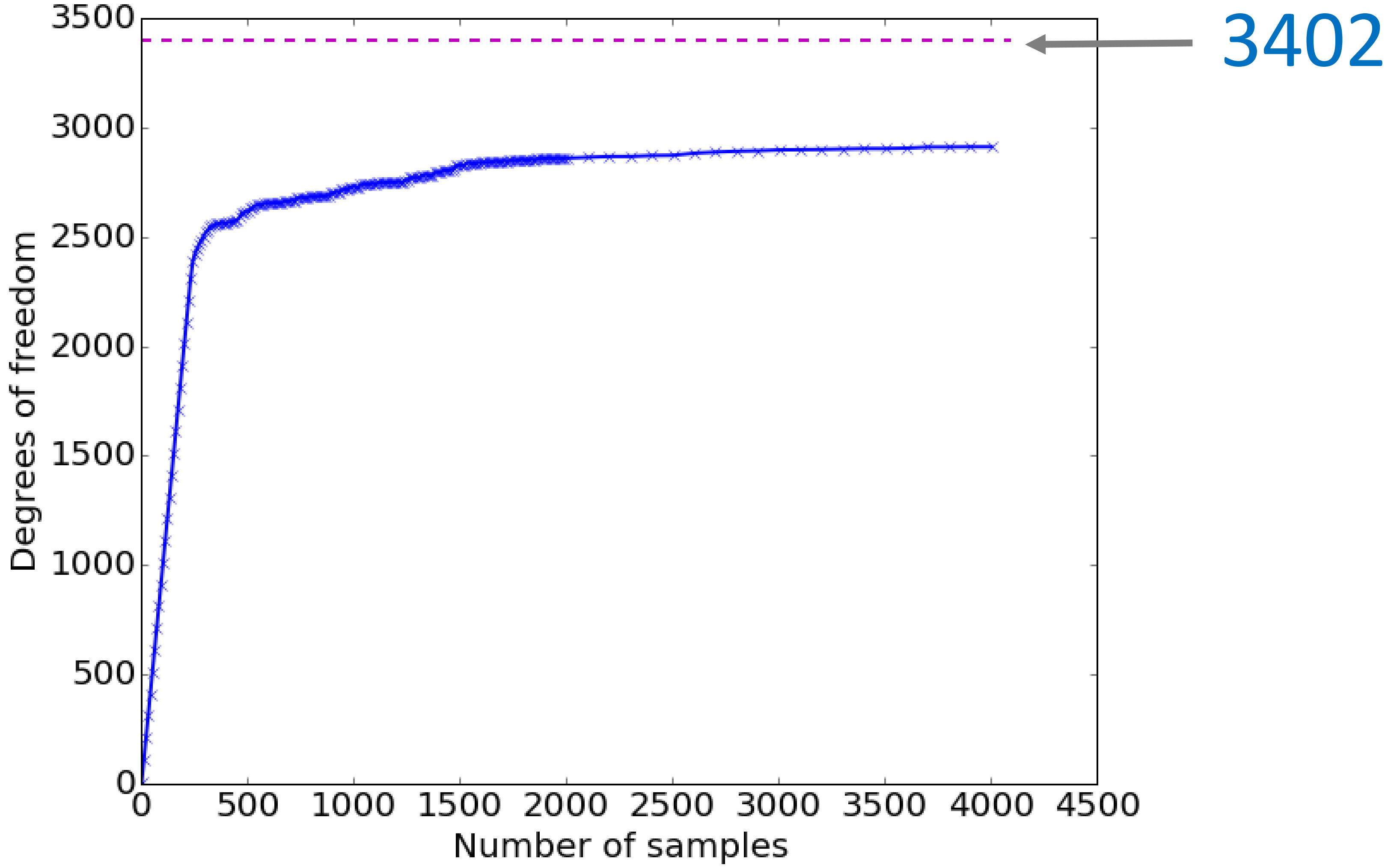
How many ambiguities do we have?

- Theorem:

Number of parameters = "degrees of freedom" + "number of ambiguities"

3466 3402? 64?

Rank of the Jacobian matrix



Discussion

- How do we remove the parameter (w) dependency?
  - Certainly there are degenerate parameter configurations
  - Is there a typical behavior?
  - Large scale limist?
- How do we remove the input dependency?
  - Can we separate the property of the network (DOF) from the property of the input distribution?

References

- Neyshabur, Tomioka, Srebro (2015) “In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning”, ICLR.
- Neyshabur, Tomioka, Srebro (2015) “Norm-Based Capacity Control in Neural Networks”, COLT.
- Neyshabur, Salakhutdinov, Srebro (2015) “Path-SGD: Path-Normalized Optimization in Deep Neural Networks”, NIPS.
- Neyshabur, Tomioka, Srebro (2016) “Data-Dependent Path Normalization in Neural Networks, ICLR.

The Jacobian

- Let's look at the Jacobian

$$J = \begin{bmatrix} \frac{\partial f(x_1)}{\partial w_1} & \frac{\partial f(x_1)}{\partial w_2} & \cdots & \frac{\partial f(x_1)}{\partial w_d} \\ \frac{\partial f(x_2)}{\partial w_1} & \frac{\partial f(x_2)}{\partial w_2} & & \\ \vdots & & & \\ \frac{\partial f(x_n)}{\partial w_1} & & \cdots & \frac{\partial f(x_n)}{\partial w_d} \end{bmatrix}$$

Invariance

- Let w and θ be two ways to parameterize the same set of functions. Assume that there is a smooth one-to-one mapping between them.
- We say that an algorithm is invariant if

$$\Delta w = \frac{dw}{d\theta} \Delta\theta$$

Direction the algorithm chooses for parameterization w

Direction the algorithm chooses for parameterization θ