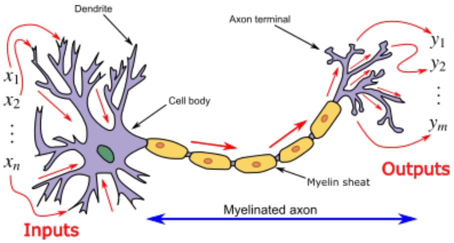
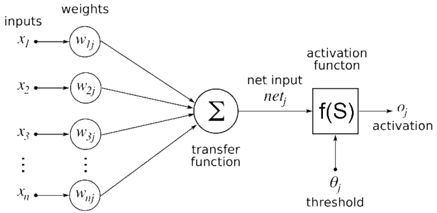
Basic concepts of artificial neural networks

Neural network inspiration



(a) Neuron in biological neural network



(b) Neuron in artificial neural network

Neural network inspiration

Biological Neural Network (BNN)	Artificial Neural Network (ANN)
Soma (Neuron body)	Node
Dendrites	Input
Synapse	Weights or Interconnections
Axon	Output

Neural network as a graph

Neurons

Let $u, v \in \mathcal{V}$ are neurons represented as vertices of a graph.

Connection links

The tuples (u, v) or (v, u) are connection links represented as oriented edges. $\mathcal{C} \subset \mathcal{V} \times \mathcal{V}$ is the set of all edges.

Neural net

$(\mathcal{V}, \mathcal{C})$ is a graph representing a neural net.

Communication with environment

- In our definition, the neurons can communicate with each other.
- We need to communicate with an environment ϖ.

Input and output connection links

$\varepsilon \subset \{\varpi\} \times \mathcal{V} \cup \mathcal{V} \times \{\varpi\}$

Input node

If $((\varpi, u) \in \varepsilon$, then the node u receives signals from the environment.

Output node

If $((v, \varpi) \in \varepsilon$, then the node v transfers signals to the environment.

Topology

- The triplet (V, C, ε) is called topology.

Input and output sets

Let us define:

- input set of neuron v:

$$i(v) : \{u : u \in \mathcal{V} \& (u, v) \in \mathcal{C}\}$$

- output set of neuron v:

$$o(v) : \{u : u \in \mathcal{V} \& (v, u) \in \mathcal{C}\}$$

Neuron types with respect to connections

- Input nodes:

$$\mathcal{I} = \{v : v \in \mathcal{V} \& i(v) = \emptyset\}$$

- Output nodes:

$$\mathcal{O} = \{v : v \in \mathcal{V} \& o(v) = \emptyset\}$$

- Hidden nodes

$$\mathcal{H} = \mathcal{V} \setminus (\mathcal{I} \cup \mathcal{O})$$

Important neural net conditions

- The graph (V, C) is non-redundant.

$$(\forall v \in \mathcal{V})(\exists u \in \mathcal{V})\{(u, v), (v, u)\} \cap \mathcal{C} \neq \emptyset$$

- A neuron can transfer a signal to other neurons only if it received a signal from one or more neurons or from the environment.
- A neuron that received a signal has to transfer a signal to other neurons or to the environment.

Neuron types with respect to connections

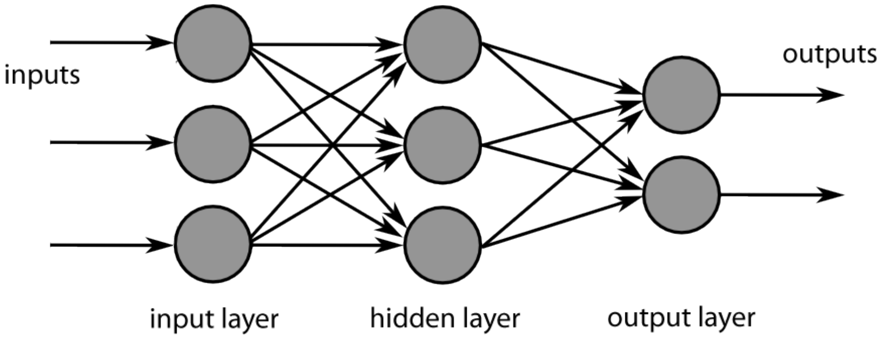


Figure: Feed forward neural network organized in layers

Neuron activity

- Time: $\mathcal{T} \subset \mathcal{R}$
- $\mathcal{T}_t^- = \mathcal{T} \cap (-\infty, t\rangle$
- We can define the *activity of a neuron v*:

$$z_v : \mathcal{T} \to \mathcal{R}$$

- The activity can have range restrictions:
  - $z_v : \mathcal{T} \to \langle 0, 1\rangle$ - normalized activity
  - $z_v : \mathcal{T} \to \langle -1, 1\rangle$
- Network state: $z(t) = (z_v(t))_{v \in \mathcal{V}}$

Global active dynamics

- At each time t the network performs a mapping F_t of input neuron activities to output neuron activities.
- We define the set of all feasible mappings $\mathcal{F}_t$
- The system $(F_t)_{t\in\mathcal{T}}$ is called *active dynamics of the network*
- Requirements:
    - The same domain for all elements
    - A finite number of parameters

$$(\exists k \in \mathcal{N})(\forall t \in \mathcal{T})(\exists D_t \subset \{\mathcal{T}_t^- \to \mathcal{R}^{|\mathcal{I}|}\})(\exists \pi_t : \mathcal{R}^k \to \{D_t \to \mathcal{R}^{|\mathcal{O}|}\})$$
$$\mathcal{F}_t = \pi_t(\mathcal{R}^k)$$

    - Restrictions on possible parameter values

$$(\exists k \in \mathcal{N})(\forall t \in \mathcal{T})(\exists W_t \subset \mathcal{R}^k)(\exists D_t \subset \{\mathcal{T}_t^- \to \mathcal{R}^{|\mathcal{I}|}\})$$
$$(\exists \pi_t : \mathcal{W}_t \to \{D_t \to \mathcal{R}^{|\mathcal{O}|}\})\mathcal{F}_t = \pi_t(W_t)$$

Local active dynamics

- System of functions $(\psi_t^v)_{t \in \mathcal{T}, v \in \mathcal{V} \setminus \mathcal{I}}$ with the following properties:
  1. For each $t \in \mathcal{T}$, each F_t can be expressed as a composition of mappings ψ_t^v that transform the activities of the input neurons $i(v), v \in \mathcal{V} \setminus \mathcal{I}$ into the activity of the neuron v at the time t.
  2. For each time t and each $v \in \mathcal{V} \setminus \mathcal{I}$, the function ψ_t^v is taken from a set Ψ_t^v of possible functions.
  3. For each time t and each $v \in \mathcal{V} \setminus \mathcal{I}$, all elements of Ψ_t^v have the same domain.

$$(\forall v \in \mathcal{V} \setminus \mathcal{I})(\exists k_v \in \mathcal{N})(\forall t \in \mathcal{T})(\exists W_t^v \subset \mathcal{R}^{k_v})$$
$$(\exists D_t^v \subset \{T_t^- \to \mathcal{R}^{|i(v)|}\})(\exists \pi_t^v : W_t^v \to \{D_t^v \to \mathcal{R}\})\Psi_t^v = \pi_t^v(W_t^v)$$

Local active dynamics II.

- We can assign each parameter to a neuron $v \in \mathcal{V} \setminus \mathcal{I}$ or to a connection $(u, v) \in \mathcal{C}$.
- An example neuron parameter: threshold θ_v.
- A usual connection parameter: connection weight $w_{(u,v)}$.
- The activity z_v of a neuron $v \in \mathcal{V} \setminus \mathcal{I}$ is often defined as:

  $$z_v(t) = f(\sum_{u \in i(v)} w_{(u,v)}(t) z_u(t) + \theta(t)),$$

  where f is a function called activation function.
- For output neurons, an identity activation function is often used.

Time-independent global active dynamics

- Time independent version is very common in practical applications
- Global active dynamics:

$$(\exists D \subset \mathcal{R}^{|\mathcal{I}|})\mathcal{F} = \{F : D \to \mathcal{R}^{|\mathcal{O}|}, \}$$

or with a parametrization:

$$(\exists k \in \mathcal{N})(\exists W \subset \mathcal{R}^{k})(\exists D \subset \mathcal{R}^{|\mathcal{I}|})(\exists \pi : W \to \{D \to \mathcal{R}^{|O|}\}) \\ \mathcal{F} = \pi(W) \quad (1)$$

Time-independent local active dynamics

- Local active dynamics:

$$(\forall v \in \mathcal{V} \setminus \mathcal{I})(\exists k_v \in \mathcal{N})(\exists W_v \subset \mathcal{R}^{k_v})(\exists D_v \subset \mathcal{R}^{|i(v)|})$$
$$(\exists \pi_v : W_v \to \{D_v \to \mathcal{R}\})\Psi_v = \pi_v(W_v)$$

- Time-independent neuron activity:

$$z_v = f(\sum_{u \in i(v)} w_{(u,v)} z_u + \theta_v)$$

Adaptive dynamics

- We have shown that the neuron activity can be time-dependent.
- Global and local dynamics can be time-dependent as well:

$$(F_\tau)_{\tau\in\mathcal{T}} \text{ , or } (\psi^v_\tau)_{\tau\in\mathcal{T}}$$

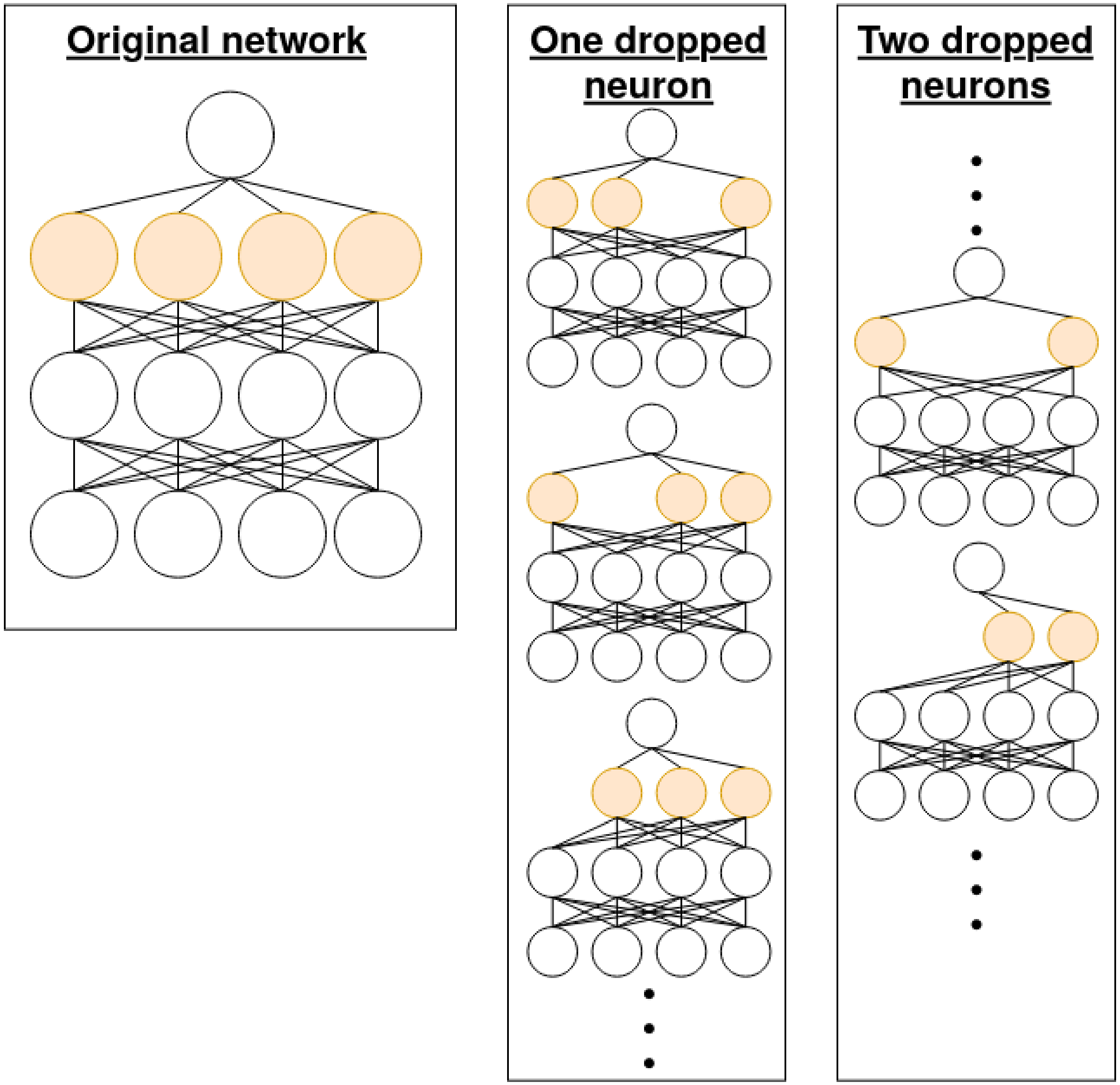
- They depend on the following factors:
  - Previous evolution of F_t, $(F_\tau)_{\tau\in\mathcal{T}_t^-\setminus\{t\}}$
  - Previous evolution and current value of neuron activities $(z_v|\mathcal{T}_t^-)_{v\in\mathcal{V}}$
  - Information from a supervisor:
    - correct (required) value that the network should output,
    - a non-negative value expressing dissimilarity of output and correct value (loss function),
    - a non-negative value expressing supervisor's satisfaction.

Loss function

- Mapping $\gamma : \mathcal{R}^{|\mathcal{O}|} \times \mathcal{R}^{|\mathcal{O}|} \rightarrow \mathcal{R}_0^+$.
- Function $\gamma(d, a)$ is called *error function* or *loss function*, where d is the correct value and a is output of the network.
- Common loss functions:
  - Sum of least squares: $\gamma(a, d) = \sum_{i=1}^{|\mathcal{O}|} |a_i - d_i|^2$
  - Cross entropy: $\gamma(a, d) = -\sum_{i=1}^{|\mathcal{O}|} \left(d_i \log a_i + (1 - d_i)\log(1 - a_i)\right)$
  - Logistic loss: $\gamma(a, d) = -da + \log(e^a + e^{-a}) = \log \frac{e^a + e^{-a}}{e^{da}}$

Dropout

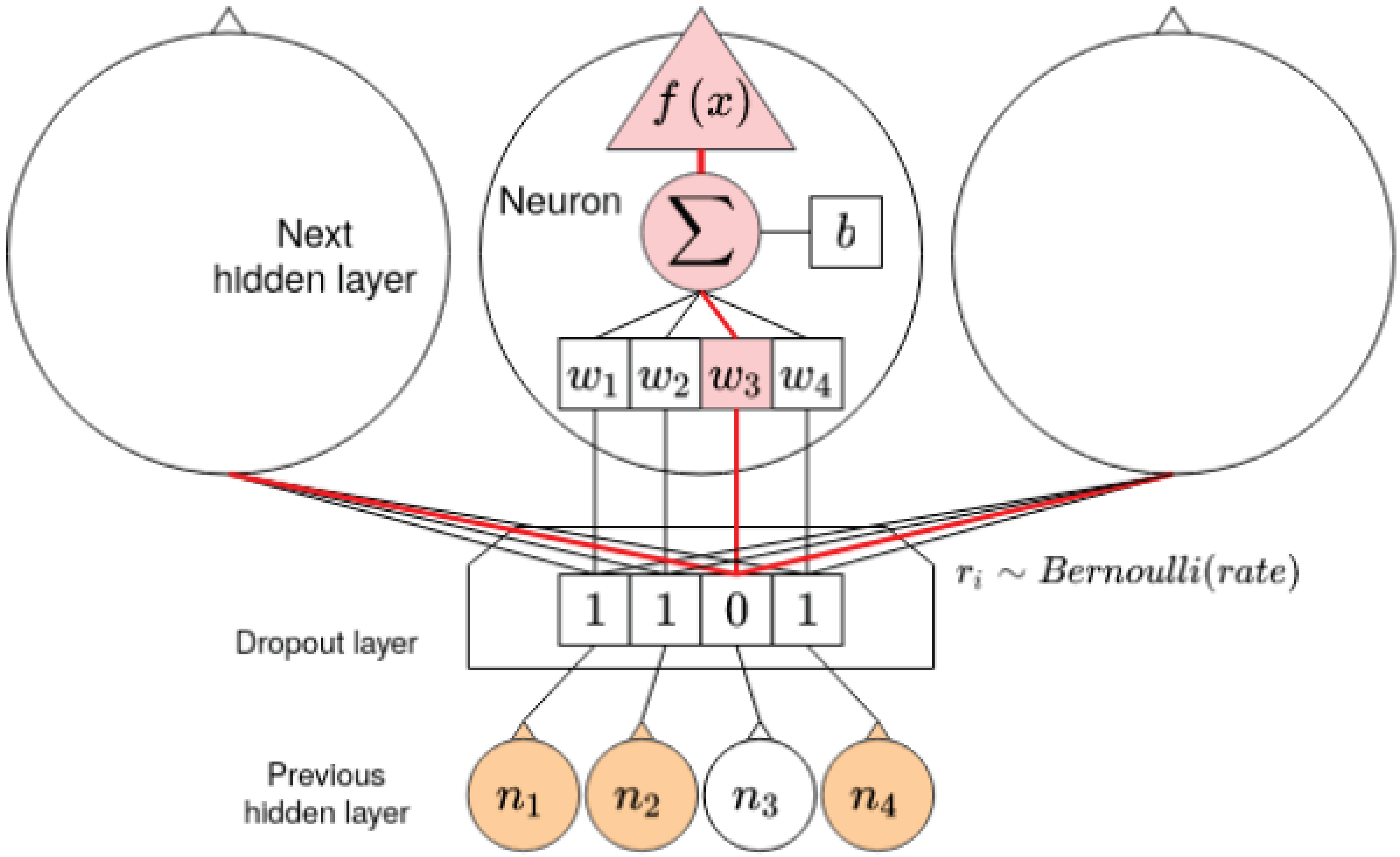
- Temporarily *removing* (dropping out) *some* input or hidden *neurons during* network *training*
- Neurons are dropped out *randomly*, according to a given distribution
- Originally proposed for and most often used during training of multilayer perceptrons



Original network
One dropped neuron
Two dropped neurons

Bernoulli dropout

- Bernoulli(p) distribution: on $\{a, b\}$ with probabilities $(1-p, p)$
- Assumptions about l-th *hidden layer*, $l = 1, \ldots, L$:
  - vectorial input $z^{(l)}$, output $y^{(l)}$, weight $w^{(l)}$, scalar bias $b^{(l)}$
  - activation function f does not depend on l, relates $y^{(l)} = f\left(z^{(l)}\right)$
  - in addition: set $a = 0, b = 1$, denote $y^{(0)} = x$ – network input
- Then $z_i^{(l)} = w_i^{(l)} r_i^{(l)} y_i^{(l-1)} + b^{(l)}$, with *random* $r_i^{(l)}$~*Bernoulli*(p)



$f(x)$
Neuron
$\sum$
b
Next hidden layer
w_1 w_2 w_3 w_4
$r_i \sim Bernoulli(rate)$
1 1 0 1
Dropout layer
Previous hidden layer
n_1 n_2 n_3 n_4

Dropout and network training

- ♦ Most often using stochastic gradient descent
- ♦ Difference from standard MLP: for each *training case*,

  new values $r_i^{(l)}$ are sampled $\Longrightarrow$ a new *specific network*

  - forward- and backpropagation restricted to that individual network
- ♦ *Gradients* are *averaged* over cases retaining the parameter
  - cases with that parameter dopped out $\Longrightarrow$ gradient contribution = 0

Dropout and regularization

- ♦ Dropout alone improves training, with regularization even more
- ♦ Most often combined with *max-norm regularization:* $\|w\| \leq c$
  - w – vector of all weights, $\|\ \ \|$ – some norm, c – hyperparameter
  - $\Longrightarrow$ network learning is then constrained optimization
- ♦ Main reason why max-norm regularization is useful:

  no weigths blowup through large learning rate $\Longrightarrow$ explorability

Some other properties of dropout

- *Sparse representation*, even if no sparsity inducing regularizers
- *Influence of dataset size* relatively to network size:
  - very small datasets overfitting even after dropout $\Longrightarrow$ useless
  - with increasing dataset size, its usefulness increases, then again decreases $\Longleftarrow$ for very large datasets, no overfitting occurs
- Training time: $2-3\times$ longer than without dropout

Advantages of dropout

1. After dropout, the network has less parameters $\Longrightarrow$

   $\Longrightarrow$ less prone to *overfitting* the training data

2. Breaking-up co-adaptations of different hidden neurons,

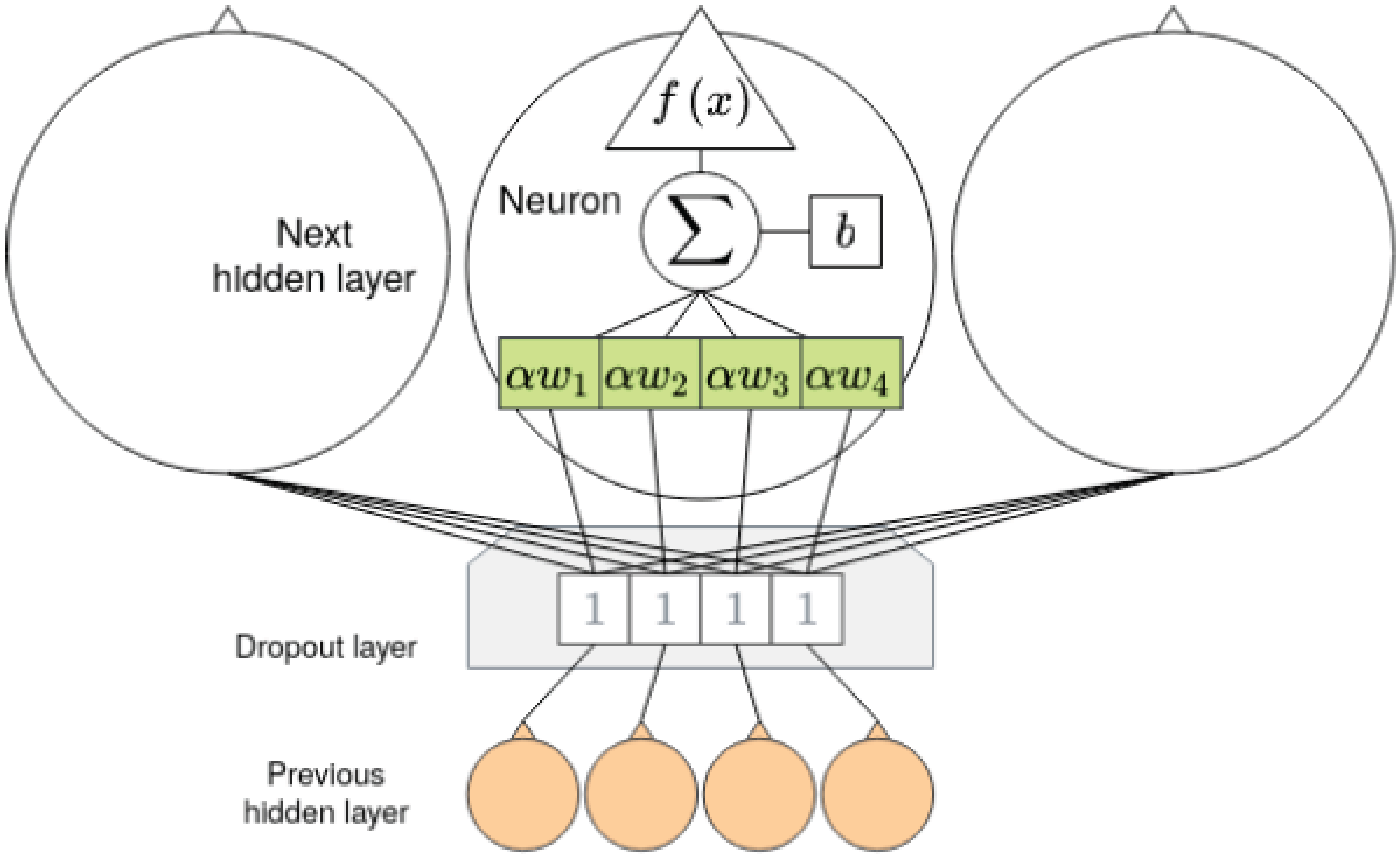
   which impede generalization $\Longrightarrow$ improved *generalization*

3. Different dropout realizations $\approx$ different network topologies $\Longrightarrow$

   $\Longrightarrow$ dropout implies building network *ensembles*

Dropout ensembles

- ♦ For an ensemble S built through dropping out subsets of the set H of hidden neurons: $|\mathcal{S}| \leq 2^{|H|}$
- ♦ If during training, $h \in H$ survives dropout with probability p, then during testing, weights outgoing from h are multiplied by p
  - $\Rightarrow$ *expected* weights *after training = used testing weights*
- ♦ Alternative possibility: training weights multilplied by $\frac{1}{p}$



$f(x)$
Neuron
$\sum$
b
Next hidden layer
αw_1
αw_2
αw_3
αw_4
1
1
1
1
Dropout layer
Previous hidden layer

More general dropouts

- Used also with other models than multilayer perceptrons
  - *restricted Boltzmann* machine (RBM, will be described later)
  - *linear regression* (will be described later)
- Used also with other distributions than Bernoulli
  - *Gaussian distribution* (will be described later)

Introducing dropout into RBM

- RBM with visible units $v \in \{0,1\}^{d_v}$, hidden units $h \in \{0,1\}^{d_h}$ and parameters $\theta = (W, a, b), W \in \mathbb{R}^{d_v \times d_h}, a \in \mathbb{R}^{d_h}, b \in \mathbb{R}^{d_v}$, which define $P(h, v; \theta) = \frac{\exp(v^\intercal W h + a^\intercal h + b^\intercal v)}{C(\theta)}, C(\theta)$ - normalizing constant
- *Dropout* is introduced with a $\{0,1\}^{d_h}$-valued random vector r with random components r_j~*Bernoulli(p)*, $r_j = 1 \Leftrightarrow h_j = 1$,
  - consequence: $r_j = 1 \Rightarrow h_j = 1, r_j = 0 \Rightarrow h_j = 0$

Dropout RBM probability distribution

- *Joint* distribution of v *and* h, with a normalizing constant $C(\theta, r)$:

$$P(h, v; \theta) = \frac{\exp(v^\intercal W h + a^\intercal h + b^\intercal v)}{C(\theta, r)} \prod_{j=1}^{d_h} \left(\mathbb{I}(r_j = 1) + \mathbb{I}(r_j = 0)\mathbb{I}(h_j = 0)\right)$$

- Conditional distribution of *h conditioned on r and v*:

$$P(h|r, v) = \prod_{j=1}^{d_h} P(h_j|r_j, v), P(h_j = 1|r_j, v) = \mathbb{I}(r_j = 1)\sigma\left(b_j + \sum_i W_{ij} v_i\right)$$

- Conditional distribution of *v on h* (same as without dropout):

$$P(v|h) = \prod_{i=1}^{d_v} P(v_i|h), P(v_i = 1|h) = \sigma\left(a_i + \sum_i W_{ij} h_j\right)$$

Dropout in linear regression

- Dropped out are individual training pairs − rows of (X, y)
  - $X \in \mathbb{R}^{N\times d}$ − matrix of N data points, $y \in \mathbb{R}^{N}$ − vector of targets
- Dropout introduced through a component-wise product $X \odot R$
  - $R \in \{0,1\}^{N\times d}$ is a $\{0,1\}^{N\times d}$-valued random matrix
  - R has all its *components random* R_{ij}*~Bernoulli*(p)

Learning dropout linear regression

- Learning in traditional linear regression consists in finding a weight vector $w \in \mathbb{R}^d$ minimizing the error $\|y - Xw\|^2$
- For dropout linear regression learning, the *minimized error* turns to $\mathbb{E}_{R\sim\mathrm{Bernoulli}(p)}\|y - X \odot R\, w\|^2 =$ (after computation)

$$= \|y - pXw\|^2 + p(1-p)\left\|\left(\mathrm{diag}(X^\top X)\right)^{\frac{1}{2}} w\right\|^2 =$$

$$= \|y - X\tilde{w}\|^2 + \frac{1-p}{p}\left\|\left(\mathrm{diag}(X^\top X)\right)^{\frac{1}{2}} \tilde{w}\right\|^2$$, with $\tilde{w} = pw$

Gaussian dropout

- Basic idea: *activation h_i* of the hidden neuron i is perturbed to $h_i(1 + r)$ with $r \sim N(0,1)$, more generally $r \sim N(0, \sigma^2)$
- Equivalently: activation h_i is *perturbed to $h_i r'$* with $r' = 1 + r$, hence $r' \sim N(1,1)$, more generally $r' \sim N(1, \sigma^2)$
- *Hyperparameter σ^2, like p* in Bernoulli dropout

What does the Gaussian drop out?

- Formally, Gaussian dropout drops no neurons out,

  only perturbs the activations of hidden neurons

- However, for $h_i r'$ with $r' \sim N(1, \sigma^2)$, where $\sigma^2 = \frac{1-p}{p}$:

  the expectation and variance of r' are $\mathbb{E}r' = 1, \mathrm{Var}\, r' = \frac{1-p}{p}$

- And the same $\mathbb{E}r'$ and $\mathrm{Var}\, r'$ has $r' \sim$ *Bernoulli*(p) on $\left\{0, \frac{1}{p}\right\}$,

  which drops out the hidden neuron i

Stochastic gradient

- Minimizing a loss function for data $x = (x_1, \dots, x_N)$, parameters θ,

  summed over data: $\mathcal{L}(\theta) = \mathcal{L}(\theta, x) = \frac{1}{N}\sum_{n\in\hat{N}} \ell_n(\theta, x_n), \hat{N} = \{1, \dots, N\}$
- For $s \in \mathbb{N}$, consider a random variable $\mathcal{M}: \{S \subset \hat{N} | \#S = s\}$*-valued*,

  called minibatch, uniformly distributed: $S \subset \hat{N} \,\&\, \#S = s \Rightarrow P(S) = \frac{1}{\binom{N}{s}}$
- Define a random loss function: $\hat{\mathcal{L}}_s(\theta) = \hat{\mathcal{L}}_s(\theta, x) = \frac{1}{s}\sum_{n\in\mathcal{M}} \ell_n(\theta, x_n)$
- Its *gradient* $\hat{g}_s = \nabla_\theta \hat{\mathcal{L}}_s$ is called *stochastic gradient*

Example: quadratic loss

- ♦ $\ell_n(\theta, x_n) = \frac{1}{2}\|x_n - \theta\|^2$, $\mathcal{L}(\theta) = \mathcal{L}(\theta, x) = \frac{1}{2N}\sum_{n=1}^{N}\|x_n - \theta\|^2$
- ♦ $\nabla_\theta \mathcal{L}(\theta, x) = \bar{x} - \theta$, with $\bar{x} = \sum_{n=1}^{N} x_n$, thus $\arg\min_\theta \mathcal{L}(\theta, x) = \bar{x}$
  - the reparametrization $\theta_{\text{new}} = \theta - \bar{x}$ leads to $\arg\min_{\theta_{\text{new}}} \mathcal{L}(\theta_{\text{new}}, x) = 0$
- ♦ $\hat{\mathcal{L}}_s(\theta) = \frac{1}{2s}\sum_{n\in\mathcal{M}}\|x_n - \theta\|^2$, $\hat{g}_s(\theta) = \nabla_\theta \hat{\mathcal{L}}_s(\theta) = \frac{1}{s}\sum_{n\in\mathcal{M}}(x_n - \theta)$

Stochastic gradient descent (SGD)

- Stochastic gradient descent is the application of *gradient descent* with learning rate ϵ to *stochastic gradient*:

$$\theta(t+1) = \theta(t) - \epsilon \hat{g}_s(\theta(t), x) = \theta(t) - \epsilon\, \nabla_\theta \hat{\mathcal{L}}_s(\theta(t), x)$$

  - $\theta(t)$ – value of the parameters in the iteration t
  - $\hat{g}_s(\theta, x) = \nabla_\theta \hat{\mathcal{L}}_s(\theta, x) = \frac{1}{s} \sum_{n \in \mathcal{M}} \nabla_\theta \ell_n(\theta, x_n)$

- SGD is studied using 4 generally accepted assumptions

Assumption 1

- Conditioned on $\theta, x_1, \dots, x_n$ *are conditionally independent identically distributed* and such that $\hat{g}_s(\theta, x) = \frac{1}{s}\sum_{n\in\mathcal{M}} \nabla_\theta \ell_n(\theta, x_n)$ behaves like $\nabla_\theta \ell_n(\theta, x_n)$ were normal random variables: $\nabla_\theta \ell_n(\theta, x_n) \sim N\left(g(\theta), C(\theta)\right)$
- $\Longrightarrow \hat{g}_s(\theta, x) - g(\theta) = \frac{1}{s}\sum_{n\in\mathcal{M}}\left(\nabla_\theta \ell_n(\theta, x_n) - g(\theta)\right) \sim N\left(0, \frac{1}{s}C(\theta)\right)$
- $\Longrightarrow$ defining $\Delta g(\theta,x) = \sqrt{s}\left(\hat{g}_s(\theta,x) - g(\theta)\right)$ implies $\Delta g(\theta,x) \sim N\left(0, C(\theta)\right)$

Assumption 2

- In a *neighborhood Θ of* an *minimal $\mathcal{L}(\theta)$* is *$C(\theta)$ constant* and *positive-definite*: ∃ a positive-definite C $\forall\theta \in \Theta: C(\theta) = C$
- Positive definiteness of $C \implies \exists$ a regular matrix $B: C = BB^\intercal$
- Defining $\Delta\theta(t) = \theta(t+1) - \theta(t), \Delta w(\theta, x) = -B^{-1}\Delta g(\theta, x)$ implies:

1. $\Delta\theta(t) = -\epsilon g\big(\theta(t)\big) - \frac{\epsilon}{\sqrt{s}}\Delta g(\theta(t), x) = -\epsilon g\big(\theta(t)\big) + \frac{\epsilon}{\sqrt{s}} B \Delta w(\theta(t), x)$
2. $\Delta w(\theta, x) \sim N(0, -B^{-1}C(\theta)(-B^\intercal)^{-1}) = N(0, B^{-1}BB^\intercal(B^\intercal)^{-1}) = N(0, I)$

Assumption 3

- The equation $\Delta\theta(t) = -\epsilon g\big(\theta(t)\big) + \frac{\epsilon}{\sqrt{s}} B\Delta w(\theta(t), x),\ \Delta w(\theta, x) \sim N(0, I)$,

  which is a finite-difference equation for $\Delta\theta(t) = \theta(t+1) - \theta(t)$,

  is replaceable with a differential equation: $\frac{d\theta}{dt} = -\epsilon g(\theta) + \frac{\epsilon}{\sqrt{s}} B \frac{dw}{dt}$

- The equation is particularly simple for $\mathcal{L}(\theta) = \frac{1}{2}\theta^\intercal A\theta$

  $\Longrightarrow g(\theta) = A\theta \Longrightarrow \frac{d\theta}{dt} = -\epsilon A\theta + \frac{\epsilon}{\sqrt{s}} B \frac{dw}{dt}$ – Ornstein-Uhlenbeck

Example: quadratic loss

- $\ell_n(\theta, x_n) = \frac{1}{2}\|x_n - \theta\|^2$, $\mathcal{L}(\theta) = \mathcal{L}(\theta, x) = \frac{1}{2N}\sum_{n=1}^{N}\|x_n - \theta\|^2$
- $\nabla_\theta \mathcal{L}(\theta, x) = \bar{x} - \theta$, with $\bar{x} = \sum_{n=1}^{N} x_n$, thus $\arg\min_\theta \mathcal{L}(\theta, x) = \bar{x}$
  - the reparametrization $\theta_{\text{new}} = \theta - \bar{x}$ leads to $\arg\min_{\theta_{\text{new}}} \mathcal{L}(\theta_{\text{new}}, x) = 0$
- $\hat{\mathcal{L}}_s(\theta) = \frac{1}{2s}\sum_{n\in\mathcal{M}}\|x_n - \theta\|^2$, $\hat{g}_s(\theta) = \nabla_\theta \hat{\mathcal{L}}_s(\theta) = \frac{1}{s}\sum_{n\in\mathcal{M}}(x_n - \theta)$
- The covariance of x_n is $\frac{1}{s}C(\theta) = \frac{1}{s}\mathbb{E}[(x_n - \bar{x})(x_n - \bar{x})^\intercal]$
- The Hessian of $\mathcal{L}(\theta)$ is the identity matrix, $A = I$

Assumption 4

- The *loss* function is *on Θ quadratic*
- If for the original parameters $\mathcal{L}(\theta') = \theta'^{\top} A' \theta' + b\theta' + c'$, then

  the transformation $\theta = \theta' + \frac{1}{2} A'^{-1} b, A = 2A', c = c' - \frac{1}{4} b^{\top} {A'^{-1}}^{\top} A'^{-1} b$

  yields $\mathcal{L}(\theta) = \frac{1}{2} \theta^{\top} A \theta + c \Longrightarrow g(\theta) = A\theta \Longrightarrow \frac{d\theta}{dt} = -\epsilon A \theta + \frac{\epsilon}{\sqrt{s}} B \frac{dw}{dt}$
  - the solution of this differential equation is

    a random process called *Ornstein-Uhlenbeck process*