Lecture 9: Plasticity as optimization

Samuel Gershman

Harvard University

Roadmap

- Neural plasticity refers broadly to adaptive change over time in the brain.

Roadmap

- Neural plasticity refers broadly to adaptive change over time in the brain.
- Much of the work on neural plasticity has focused on synapses as the site of learning and memory. This lecture conceptualizes synaptic plasticity in terms of optimization: modifications of synaptic strength lead to improvements in performance (as measured by an objective function).

Roadmap

- Neural plasticity refers broadly to adaptive change over time in the brain.
- Much of the work on neural plasticity has focused on synapses as the site of learning and memory. This lecture conceptualizes synaptic plasticity in terms of optimization: modifications of synaptic strength lead to improvements in performance (as measured by an objective function).
- The most efficient algorithms for improving performance are based on gradient descent, delivering vector feedback that assigns credit to individual synapses. We will discuss ways that synaptic plasticity might approximate gradient descent, either explicitly or implicitly.

The computational problem

- Given an objective function, how can the brain adapt to efficiently optimize the objective?

The computational problem

- Given an objective function, how can the brain adapt to efficiently optimize the objective?
- Efficiency means several things in this context, including sample complexity (how much data are needed to achieve a particular level of performance?), time complexity (how much computation is required as a function of network size and other parameters?), and space complexity (how much information needs to be stored in order to implement the algorithm?).

The computational problem

- Given an objective function, how can the brain adapt to efficiently optimize the objective?
- Efficiency means several things in this context, including sample complexity (how much data are needed to achieve a particular level of performance?), time complexity (how much computation is required as a function of network size and other parameters?), and space complexity (how much information needs to be stored in order to implement the algorithm?).
- Another important consideration is scalability: how well does the algorithm work in practice on realistically large-scale problems?

The computational problem

- Given an objective function, how can the brain adapt to efficiently optimize the objective?
- Efficiency means several things in this context, including sample complexity (how much data are needed to achieve a particular level of performance?), time complexity (how much computation is required as a function of network size and other parameters?), and space complexity (how much information needs to be stored in order to implement the algorithm?).
- Another important consideration is scalability: how well does the algorithm work in practice on realistically large-scale problems?
- Biological constraints: can synapses implement the algorithm based only on the information available to it? Will the algorithm work for spiking neurons?

A brief tour of synaptic plasticity

- Much of what we know about synaptic plasticity comes from protocols in which the axon of a presynaptic neuron is electrically stimulated while recording from its postsynaptic partner.

A brief tour of synaptic plasticity

- Much of what we know about synaptic plasticity comes from protocols in which the axon of a presynaptic neuron is electrically stimulated while recording from its postsynaptic partner.
- At baseline, a brief pulse of electrical stimulation induces a measurable change in the membrane potential of the postsynaptic neuron—an *excitatory postsynaptic potential* (EPSP).

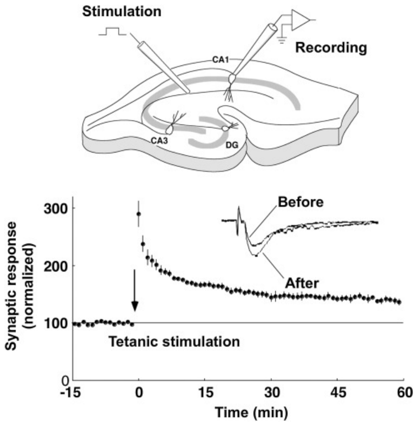
A brief tour of synaptic plasticity

- Much of what we know about synaptic plasticity comes from protocols in which the axon of a presynaptic neuron is electrically stimulated while recording from its postsynaptic partner.
- At baseline, a brief pulse of electrical stimulation induces a measurable change in the membrane potential of the postsynaptic neuron—an *excitatory postsynaptic potential* (EPSP).
- A high-frequency train of stimulation is then applied, and the EPSP in response to the same test pulse used at baseline is measured.

A brief tour of synaptic plasticity

- Much of what we know about synaptic plasticity comes from protocols in which the axon of a presynaptic neuron is electrically stimulated while recording from its postsynaptic partner.
- At baseline, a brief pulse of electrical stimulation induces a measurable change in the membrane potential of the postsynaptic neuron—an *excitatory postsynaptic potential* (EPSP).
- A high-frequency train of stimulation is then applied, and the EPSP in response to the same test pulse used at baseline is measured.
- *Long-term potentiation* (LTP): resulting EPSP is higher compared to baseline, indicating an increase in synaptic strength.

Long-term potentiation



[Hayashi 2022]

Cellular changes underlying LTP

- At relatively short timescales (minutes to hours), there are modifications of existing receptors, such as phosphorylation of AMPA receptors by protein kinases, converting them to a high-conductance state.

Cellular changes underlying LTP

- At relatively short timescales (minutes to hours), there are modifications of existing receptors, such as phosphorylation of AMPA receptors by protein kinases, converting them to a high-conductance state.
- At relatively long timescales (hours to days), signaling cascades reach the nucleus, activating gene expression that ultimately results in the synthesis of plasticity-related proteins and the trafficking of receptors to the postsynaptic membrane.

Associative view of synaptic plasticity

- Hebb's postulate: "neurons that fire together, wire together."

Associative view of synaptic plasticity

- Hebb's postulate: "neurons that fire together, wire together."
- Coactivation rule:

$$\Delta w \propto xy$$

  where x is the presynaptic firing rate, y is the postsynaptic firing rate, and w is the synaptic strength connecting the two neurons, such that y is a monotonically increasing function of the product wx.

Associative view of synaptic plasticity

- Hebb's postulate: "neurons that fire together, wire together."
- Coactivation rule:

  $$\Delta w \propto xy$$

  where x is the presynaptic firing rate, y is the postsynaptic firing rate, and w is the synaptic strength connecting the two neurons, such that y is a monotonically increasing function of the product wx.
- Empirically, both presynaptic and postsynaptic activity are necessary to induce LTP.

Associative view of synaptic plasticity

- Associative nature of LTP fits broadly with associative models of learning: changes in behavior during associative learning tasks arise from changes in associations between stimuli.

Associative view of synaptic plasticity

- Associative nature of LTP fits broadly with associative models of learning: changes in behavior during associative learning tasks arise from changes in associations between stimuli.
- For example, in a Pavlovian fear conditioning task, an animal is repeatedly exposed to a neutral stimulus (e.g., tone) paired with an aversive stimulus (e.g., footshock). The animal learns to produce a conditioned response (freezing) when it hears the tone, a process that depends on the amygdala.

Associative view of synaptic plasticity

- Associative nature of LTP fits broadly with associative models of learning: changes in behavior during associative learning tasks arise from changes in associations between stimuli.
- For example, in a Pavlovian fear conditioning task, an animal is repeatedly exposed to a neutral stimulus (e.g., tone) paired with an aversive stimulus (e.g., footshock). The animal learns to produce a conditioned response (freezing) when it hears the tone, a process that depends on the amygdala.
- The conditioned response can be inactivated by reducing synaptic strength at amygdala synapses, and reactivated by increasing it [Nabavi et al 2014], indicating that synaptic plasticity plays a causal role in associative learning.

Spike timing-dependent plasticity

- Presynaptic spikes must occur within 50 ms *before* postsynaptic spikes in order to produce LTP.

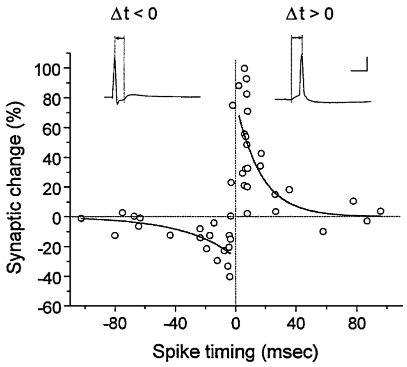
Spike timing-dependent plasticity

- Presynaptic spikes must occur within 50 ms *before* postsynaptic spikes in order to produce LTP.
- If they instead occur within 50 ms *after* postsynaptic spikes, a reduction in synaptic strength, known as *long-term depression* (LTD), is obtained.

Spike timing-dependent plasticity

- Presynaptic spikes must occur within 50 ms *before* postsynaptic spikes in order to produce LTP.
- If they instead occur within 50 ms *after* postsynaptic spikes, a reduction in synaptic strength, known as *long-term depression* (LTD), is obtained.
- Outside this ± 50 ms window, the synaptic strength does not change at all.

Spike timing-dependent plasticity



[Bi & Poo 2001]

The Graupner-Brunel model

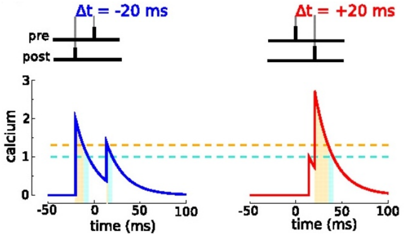
- Key idea: calcium levels that exceed a depression threshold induce LTD, and levels exceeding a higher potentiation threshold induce LTP.

The Graupner-Brunel model

- Key idea: calcium levels that exceed a depression threshold induce LTD, and levels exceeding a higher potentiation threshold induce LTP.
- Calcium transients produced by weak inputs are able to cross the depression threshold, but require larger potential changes produced by postsynaptic spiking in order to cross the potentiation threshold.

The Graupner-Brunel model

Calcium dynamics underlying STDP. The turquoise line shows the depression threshold, and the orange line shows the potentiation threshold. Δt denotes the temporal gap between presynaptic and postsynaptic spikes. The shaded regions show the time spent above each threshold.



[Graupner & Brunel 2012]

The Graupner-Brunel model

- The Graupner-Brunel model also helps resolve a fundamental problem with the basic Hebb rule: synaptic strength can potentially increase without bound, which is obviously problematic for biological synapses.

The Graupner-Brunel model

- The Graupner-Brunel model also helps resolve a fundamental problem with the basic Hebb rule: synaptic strength can potentially increase without bound, which is obviously problematic for biological synapses.
- The postulate that low levels of synaptic activity produce LTD, while high levels produce LTP, provides an important stabilizing force.

The BCM model

- Plasticity thresholds do not automatically prevent runaway plasticity; it is necessary for the thresholds to grow more quickly than the changes in postsynaptic firing rate.

The BCM model

- Plasticity thresholds do not automatically prevent runaway plasticity; it is necessary for the thresholds to grow more quickly than the changes in postsynaptic firing rate.
- Sliding thresholds, updated as a supralinear function of the firing rate [Bienenstock et al 1982]:

  $$\Delta w \propto x(y - \theta)$$

  where θ is a plasticity threshold updated according to a quadratic function of the firing rate:

  $$\Delta\theta \propto y^2 - \theta$$

The BCM model

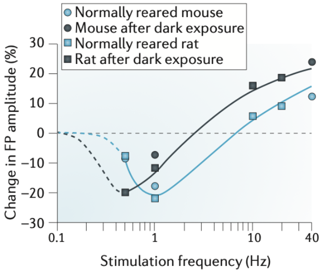
- Evidence for a sliding threshold comes from experiments on visual deprivation: a rodent reared in the dark will have weaker activation of neurons in visual cortex, and therefore is predicted to have a lower threshold compared to normally reared rodents.

The BCM model

- Evidence for a sliding threshold comes from experiments on visual deprivation: a rodent reared in the dark will have weaker activation of neurons in visual cortex, and therefore is predicted to have a lower threshold compared to normally reared rodents.
- Consistent with this hypothesis, LTP can be induced with weaker stimulation in dark-reared rodents.

Sliding plasticity threshold in visual cortex

LTP threshold is reduced in visual cortex after light deprivation.



[Cooper & Bear 2012; Kirkwood et al 1996; Philpot et al 2007]

Three-factor rules

- Another shortcoming of the basic Hebb rule is that it assumes a strongly local form of synaptic plasticity; in fact, synaptic plasticity depends on additional variables, notably neuromodulators like dopamine, serotonin, and norepinephrine.

Three-factor rules

- Another shortcoming of the basic Hebb rule is that it assumes a strongly local form of synaptic plasticity; in fact, synaptic plasticity depends on additional variables, notably neuromodulators like dopamine, serotonin, and norepinephrine.
- This leads to “three-factor” rules of the following form with neuromodulator ρ:
$$\Delta w \propto xy\rho$$

Three-factor rules

- Another shortcoming of the basic Hebb rule is that it assumes a strongly local form of synaptic plasticity; in fact, synaptic plasticity depends on additional variables, notably neuromodulators like dopamine, serotonin, and norepinephrine.
- This leads to "three-factor" rules of the following form with neuromodulator ρ:

$$\Delta w \propto xy\rho$$

- We will see how three-factor rules arise from normative considerations. We will also see how the third factor might not be a simple scalar signal, as is usually assumed in the case of neuromodulators, but might instead be vector-valued, conveyed by feedback projections.

Optimization and the credit assignment problem

- I will now lay out what the goal of synaptic plasticity might be, and how this goal can be achieved algorithmically.

Optimization and the credit assignment problem

- I will now lay out what the goal of synaptic plasticity might be, and how this goal can be achieved algorithmically.
- Central claim: synaptic plasticity optimizes an objective (also known as the *loss* or *error*) function, which scores how well the network is doing on some task.

Optimization and the credit assignment problem

- I will now lay out what the goal of synaptic plasticity might be, and how this goal can be achieved algorithmically.
- Central claim: synaptic plasticity optimizes an objective (also known as the *loss* or *error*) function, which scores how well the network is doing on some task.
- Core problem for any optimization system: *credit assignment*. Given a scalar score, how does the system know which parameter to change, and what should the parameter be changed to?

Optimization and the credit assignment problem

- I will now lay out what the goal of synaptic plasticity might be, and how this goal can be achieved algorithmically.
- Central claim: synaptic plasticity optimizes an objective (also known as the *loss* or *error*) function, which scores how well the network is doing on some task.
- Core problem for any optimization system: *credit assignment*. Given a scalar score, how does the system know which parameter to change, and what should the parameter be changed to?
- In practice, many parameters may need to be changed simultaneously, prohibiting a brute force trial-and-error approach.

Objective functions

- Denote the objective function by L, which depends on the network outputs y and an instructive signal. The network outputs are in turn dependent on parameters w.

Objective functions

- Denote the objective function by L, which depends on the network outputs y and an instructive signal. The network outputs are in turn dependent on parameters w.
- We will take w to be a set of synaptic weights, but more generally this can include any parameters governing the neural network.

Objective functions

- Denote the objective function by L, which depends on the network outputs y and an instructive signal. The network outputs are in turn dependent on parameters w.
- We will take w to be a set of synaptic weights, but more generally this can include any parameters governing the neural network.
- We will use $y = \phi_w(x)$ to denote the input-output mapping, where x denotes the inputs and y denotes the outputs.

Objective functions

Broadly speaking, objective functions fall into three categories:

- **Supervised objectives**: instructive signal takes the form of a target (or "label") vector y^*. For example, a commonly used objective is the squared error: $L(y, y^*) = \sum_i (y_i - y_i^*)^2$.

Objective functions

Broadly speaking, objective functions fall into three categories:

- **Supervised objectives**: instructive signal takes the form of a target (or "label") vector y^*. For example, a commonly used objective is the squared error: $L(y, y^*) = \sum_i (y_i - y_i^*)^2$.
- **Unsupervised objectives**: evaluate a network based on how well it matches the input data. A typical example is an autoencoder network that maps the input data to outputs of the same domain (i.e., the inputs are the target outputs, $y^* = x$); the objective function in this case corresponds to a "reconstruction error" such as squared error, $L(y, x) = \frac{1}{2} \sum_i (y_i - x_i)^2$.

Objective functions

Broadly speaking, objective functions fall into three categories:

- **Supervised objectives**: instructive signal takes the form of a target (or "label") vector y^*. For example, a commonly used objective is the squared error: $L(y, y^*) = \sum_i (y_i - y_i^*)^2$.
- **Unsupervised objectives**: evaluate a network based on how well it matches the input data. A typical example is an autoencoder network that maps the input data to outputs of the same domain (i.e., the inputs are the target outputs, $y^* = x$); the objective function in this case corresponds to a "reconstruction error" such as squared error, $L(y, x) = \frac{1}{2} \sum_i (y_i - x_i)^2$.
- **Reward objectives**: evaluate a network based on how well it predicts/improves a scalar reward (or cost). This leads to reinforcement learning algorithms.

Risk

- The agent is exposed to training data $\{x_m, y_m^*\}_{m=1}^{M}$ sampled from a distribution $p(x, y^*)$, and the objective function is evaluated on this dataset.

Risk

- The agent is exposed to training data $\{x_m, y^*_m\}_{m=1}^M$ sampled from a distribution $p(x, y^*)$, and the objective function is evaluated on this dataset.
- The agent is thus optimizing a random variable, the *empirical risk*:

$$\hat{L}(w) = \frac{1}{M} \sum_{m=1}^{M} L(\phi_w(x_m), y^*_m)$$

Risk

- The agent is exposed to training data $\{x_m, y_m^*\}_{m=1}^M$ sampled from a distribution $p(x, y^*)$, and the objective function is evaluated on this dataset.
- The agent is thus optimizing a random variable, the *empirical risk*:

$$\hat{L}(w) = \frac{1}{M} \sum_{m=1}^{M} L(\phi_w(x_m), y_m^*)$$

- Optimizing the expectation of this random variable (the *risk*, $\mathbb{E}[\hat{L}(w)]$) is typically the ultimate goal, but the agent does not have direct access to this expectation.

Inductive bias and regularization

- Because the empirical risk is a noisy estimate of the risk, optimizing it directly can lead to suboptimal generalization. Need to introduce an inductive bias that prevents the network from overfitting the data.

Inductive bias and regularization

- Because the empirical risk is a noisy estimate of the risk, optimizing it directly can lead to suboptimal generalization. Need to introduce an inductive bias that prevents the network from overfitting the data.
- Solution: optimize *regularized* empirical risk functions of the form $E(w) = \hat{L}(w) + \Omega(w)$, where $\Omega(w)$ is a regularization function.

Inductive bias and regularization

- Because the empirical risk is a noisy estimate of the risk, optimizing it directly can lead to suboptimal generalization. Need to introduce an inductive bias that prevents the network from overfitting the data.
- Solution: optimize *regularized* empirical risk functions of the form $E(w) = \hat{L}(w) + \Omega(w)$, where $\Omega(w)$ is a regularization function.
- Example: L2 regularization penalizes the Euclidean norm of the weights, $\Omega(w) \propto ||w||^2$ (intuitively, large weights are penalized).

Inductive bias and regularization

- Because the empirical risk is a noisy estimate of the risk, optimizing it directly can lead to suboptimal generalization. Need to introduce an inductive bias that prevents the network from overfitting the data.
- Solution: optimize *regularized* empirical risk functions of the form $E(w) = \hat{L}(w) + \Omega(w)$, where $\Omega(w)$ is a regularization function.
- Example: L2 regularization penalizes the Euclidean norm of the weights, $\Omega(w) \propto ||w||^2$ (intuitively, large weights are penalized).
- Theoretical analyses of generalization typically require some inductive bias in order to guarantee bounded error.

Why simple perturbation methods don't work well

- One could try randomly perturbing the synaptic weights, evaluating the objective function, and then accepting or rejecting the perturbed weights based on whether they improve the objective.

Why simple perturbation methods don't work well

- One could try randomly perturbing the synaptic weights, evaluating the objective function, and then accepting or rejecting the perturbed weights based on whether they improve the objective.
- While this can work in principle, it's hopelessly inefficient: random perturbations to a large network are vanishingly unlikely to improve the objective.

Why simple perturbation methods don't work well

- One could try randomly perturbing the synaptic weights, evaluating the objective function, and then accepting or rejecting the perturbed weights based on whether they improve the objective.
- While this can work in principle, it's hopelessly inefficient: random perturbations to a large network are vanishingly unlikely to improve the objective.
- They might be good models for evolution, but they're probably not good models for learning in the brain (at least in their simplest forms).

Gradient descent

- More efficient approach is to follow the gradient of the objective function, which specifies the direction of steepest descent in the parameter space:

$$\Delta w \propto -\nabla_w E$$

Gradient descent

- More efficient approach is to follow the gradient of the objective function, which specifies the direction of steepest descent in the parameter space:

$$\Delta w \propto -\nabla_w E$$

- In practice, it is often desirable to compute the gradient on a subset of examples. Repeatedly sampling subsets and applying the weight update is known as *stochastic gradient descent*.

Perturbation methods revisited

- Again randomly perturb the weights, but instead of accepting or rejecting the new weights, use the perturbation to estimate the gradient:

$$\nabla_w E = \frac{1}{\sigma^2}\mathbb{E}[(E(\tilde{w}) - E(w))(\tilde{w} - w)]$$

  where $\tilde{w} \sim \mathcal{N}(w, \sigma^2)$ denotes the perturbed weights.

Perturbation methods revisited

- Again randomly perturb the weights, but instead of accepting or rejecting the new weights, use the perturbation to estimate the gradient:

$$\nabla_w E = \frac{1}{\sigma^2}\mathbb{E}[(E(\tilde{w}) - E(w))(\tilde{w} - w)]$$

  where $\tilde{w} \sim \mathcal{N}(w, \sigma^2)$ denotes the perturbed weights.
- Stochastic approximation algorithm that uses random perturbations to update the weights:

$$\Delta w \propto -\frac{1}{\sigma^2}(E(\tilde{w}) - E(w))(\tilde{w} - w).$$

Perturbation methods revisited

- Again randomly perturb the weights, but instead of accepting or rejecting the new weights, use the perturbation to estimate the gradient:

$$\nabla_w E = \frac{1}{\sigma^2}\mathbb{E}[(E(\tilde{w}) - E(w))(\tilde{w} - w)]$$

  where $\tilde{w} \sim \mathcal{N}(w, \sigma^2)$ denotes the perturbed weights.

- Stochastic approximation algorithm that uses random perturbations to update the weights:

$$\Delta w \propto -\frac{1}{\sigma^2}(E(\tilde{w}) - E(w))(\tilde{w} - w).$$

- The expected update for this algorithm is equal to the exact gradient descent update.

Perturbation methods revisited

- Synapses are known to be highly unreliable: an action potential typically produces neurotransmitter release less than half of the time. Why would a neuron go to the trouble of spiking if the signal frequently fails to be propagated?

Perturbation methods revisited

- Synapses are known to be highly unreliable: an action potential typically produces neurotransmitter release less than half of the time. Why would a neuron go to the trouble of spiking if the signal frequently fails to be propagated?
- One answer [Seung 2003]: stochastic release provides information about the gradient through perturbation.

Perturbation methods revisited

- Synapses are known to be highly unreliable: an action potential typically produces neurotransmitter release less than half of the time. Why would a neuron go to the trouble of spiking if the signal frequently fails to be propagated?
- One answer [Seung 2003]: stochastic release provides information about the gradient through perturbation.
- If we think of σ^2 as a proxy for synaptic unreliability, changes in synaptic strength should be bigger for more unreliable synapses, consistent with experimental data [Bolshakov & Siegelbaum 1995].

Perturbation methods revisited

- While weight perturbation “works” by providing an unbiased estimate of the gradient, this estimate can have extremely high variance, limiting its practical usefulness—learning via weight perturbation can be orders of magnitude slower than exact gradient descent.

Perturbation methods revisited

- While weight perturbation “works” by providing an unbiased estimate of the gradient, this estimate can have extremely high variance, limiting its practical usefulness—learning via weight perturbation can be orders of magnitude slower than exact gradient descent.
- An alternative, *node perturbation*, follows the same logic, but perturbs neural activity rather than the weights, taking advantage of the stochastic nature of neural activity [Mazzoni et al 1991].

Perturbation methods revisited

- While weight perturbation "works" by providing an unbiased estimate of the gradient, this estimate can have extremely high variance, limiting its practical usefulness—learning via weight perturbation can be orders of magnitude slower than exact gradient descent.
- An alternative, *node perturbation*, follows the same logic, but perturbs neural activity rather than the weights, taking advantage of the stochastic nature of neural activity [Mazzoni et al 1991].
- Because the dimensionality of this perturbation is smaller (there are fewer neurons than synapses), it tends to have lower variance. However, this method is still suffers from high variance compared to direct computation of the gradient.

Backpropagation

- Instead of using finite difference approximations, most engineers directly compute the gradient by utilizing the chain rule of calculus.

Backpropagation

- Instead of using finite difference approximations, most engineers directly compute the gradient by utilizing the chain rule of calculus.
- This is particularly apt for deep neural networks, where computations are arranged in chains.

Backpropagation

- Instead of using finite difference approximations, most engineers directly compute the gradient by utilizing the chain rule of calculus.
- This is particularly apt for deep neural networks, where computations are arranged in chains.
- The backpropagation algorithm [Rumelhart et al 1986] is essentially an application of the chain rule to deep neural networks which enables efficient computation of gradients at all layers of the network. It is the workhorse of modern machine learning.

Backpropagation

- Key idea: recursively compute gradients, starting at the output layer and then passing the gradients down to the next layer.

Backpropagation

- Key idea: recursively compute gradients, starting at the output layer and then passing the gradients down to the next layer.
- Let $h_j = \sigma(\mu_j)$ denote the activation of neuron j, where $\sigma(\cdot)$ is an activation function and $\mu_j = \sum_i w_{ij} h_i$ is the total input to neuron j.

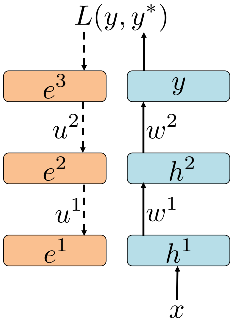
Backpropagation

- Key idea: recursively compute gradients, starting at the output layer and then passing the gradients down to the next layer.
- Let $h_j = \sigma(\mu_j)$ denote the activation of neuron j, where $\sigma(\cdot)$ is an activation function and $\mu_j = \sum_i w_{ij} h_i$ is the total input to neuron j.
- Weight update:

$$\Delta w_{ij} \propto -\frac{dE}{dw_{ij}} = -h_i e_j, \qquad e_j = \sigma'(\mu_j) \sum_k e_k w_{jk}$$

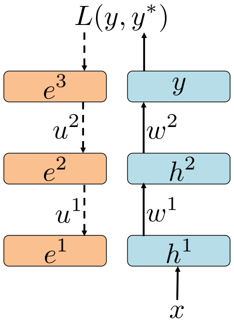
where $\sigma'(\mu_j)$ is the partial derivative of the activation function with respect to μ_j.

Backpropagation



- One way to implement backpropagation is to construct feedback connections (carrying errors) that mirror the feedforward connections (carrying activations).

Backpropagation



- One way to implement backpropagation is to construct feedback connections (carrying errors) that mirror the feedforward connections (carrying activations).
- Errors modulate a Hebbian plasticity rule, which depends on the correlation between a presynaptic term, h_i, and a postsynaptic term, $\sigma'(\mu_j)$.

Problems for biological implementations of backpropagation

- **The weight transport problem**: feedback weights need to be mirror images of the feedforward weights.

Problems for biological implementations of backpropagation

- **The weight transport problem**: feedback weights need to be mirror images of the feedforward weights.
- **The sign problem**: errors need to be signed (both positive and negative), but neural activity is always non-negative.

Problems for biological implementations of backpropagation

- **The weight transport problem**: feedback weights need to be mirror images of the feedforward weights.
- **The sign problem**: errors need to be signed (both positive and negative), but neural activity is always non-negative.
- **The magnitude problem**: errors can sometimes vary over multiple orders of magnitude, particularly in recurrent or very deep networks, producing exploding or vanishing gradients. Yet the firing rates of biological neurons typically vary over only about one order of magnitude.

Problems for biological implementations of backpropagation

- **The weight transport problem**: feedback weights need to be mirror images of the feedforward weights.
- **The sign problem**: errors need to be signed (both positive and negative), but neural activity is always non-negative.
- **The magnitude problem**: errors can sometimes vary over multiple orders of magnitude, particularly in recurrent or very deep networks, producing exploding or vanishing gradients. Yet the firing rates of biological neurons typically vary over only about one order of magnitude.
- **The update locking problem**: weight updates cannot occur in real time, because they have to wait until both the forward and backward passes have been completed.

Learning without weight symmetry

- Surprisingly, it turns out that weight symmetry is not necessary for backpropagation to be effective. In fact, backpropagation can work well even when feedback weights are random.

Learning without weight symmetry

- Surprisingly, it turns out that weight symmetry is not necessary for backpropagation to be effective. In fact, backpropagation can work well even when feedback weights are random.
- **Sign concordance**: matching the magnitudes of the feedforward and feedback weights doesn't matter much, but matching the signs matters much more.

Learning without weight symmetry

- Surprisingly, it turns out that weight symmetry is not necessary for backpropagation to be effective. In fact, backpropagation can work well even when feedback weights are random.
- **Sign concordance**: matching the magnitudes of the feedforward and feedback weights doesn't matter much, but matching the signs matters much more.
- **Feedback alignment**: with random feedback weights, backpropagation naturally adjusts the feedforward weights to achieve "soft" alignment with the feedback weights. In essence, the feedforward weights learn to partially compensate for the asymmetries created by the random feedback weights.

Dendritic segregation of feedforward and feedback signals

- Dendrites on pyramidal neurons can be divided into “apical” (emanating from the apex of the soma) and “basal” (emanating from the base of the soma).

Dendritic segregation of feedforward and feedback signals

- Dendrites on pyramidal neurons can be divided into “apical” (emanating from the apex of the soma) and “basal” (emanating from the base of the soma).
- Apical dendrites primarily receive input from feedback projections, whereas the basal dendrites primarily receive input from feedforward projections.

Dendritic segregation of feedforward and feedback signals

- Dendrites on pyramidal neurons can be divided into “apical” (emanating from the apex of the soma) and “basal” (emanating from the base of the soma).
- Apical dendrites primarily receive input from feedback projections, whereas the basal dendrites primarily receive input from feedforward projections.
- Because of their spatial separation, the activation of these different compartments are electrotonically segregated (i.e., electrical signals generated in one compartment will not passively spread to the other).

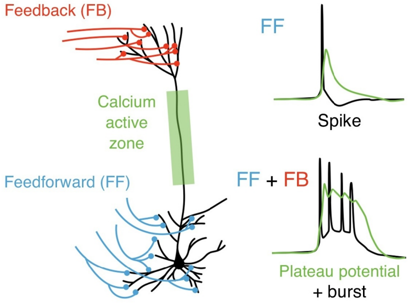
Dendritic segregation of feedforward and feedback signals

- The apical dendrites have a region dense with voltage-dependent calcium channels, which can generate "plateau potentials" (a temporally broad voltage change) when inputs to the apical dendrites coincide with somatic spiking [Larkum et al 1999].

Dendritic segregation of feedforward and feedback signals

- The apical dendrites have a region dense with voltage-dependent calcium channels, which can generate "plateau potentials" (a temporally broad voltage change) when inputs to the apical dendrites coincide with somatic spiking [Larkum et al 1999].
- This in turn produces high-frequency bursts that are required for plasticity in the basal dendrites. The bursts thus play the role of the third factor in Hebbian plasticity.

Dendritic segregation of feedforward and feedback signals



[Richards & Lillicrap 2019]

Dendritic segregation of feedforward and feedback signals

- In terms of the backpropagation algorithm, we can identify the feedforward signals with the inputs to the basal dendrites, and the error signals with the inputs to the distal apical dendrites.

Dendritic segregation of feedforward and feedback signals

- In terms of the backpropagation algorithm, we can identify the feedforward signals with the inputs to the basal dendrites, and the error signals with the inputs to the distal apical dendrites.
- Plasticity of the feedforward weights is regulated by the error signals through the cooperative generation of plateau potentials and burst spiking.

Beyond the synapse

- **Plasticity of cellular excitability**: “cell-intrinsic” in the sense that what’s changing is the cell’s overall responsiveness to its integrated inputs rather than to inputs from a specific synapse (hence it is often referred to as *intrinsic plasticity*).

Beyond the synapse

- **Plasticity of cellular excitability**: “cell-intrinsic” in the sense that what’s changing is the cell’s overall responsiveness to its integrated inputs rather than to inputs from a specific synapse (hence it is often referred to as *intrinsic plasticity*).
- For example, the same protocol that produces LTP (high-frequency stimulation of presynaptic axons) increases the probability that a postsynaptic EPSP will produce a spike. This change in EPSP-spike coupling is distinct from the synapse specific changes underlying LTP.

Beyond the synapse

- More radical proposals: neurons may learn and remember more complex forms of information using cell-intrinsic mechanisms that go beyond changes in excitability.

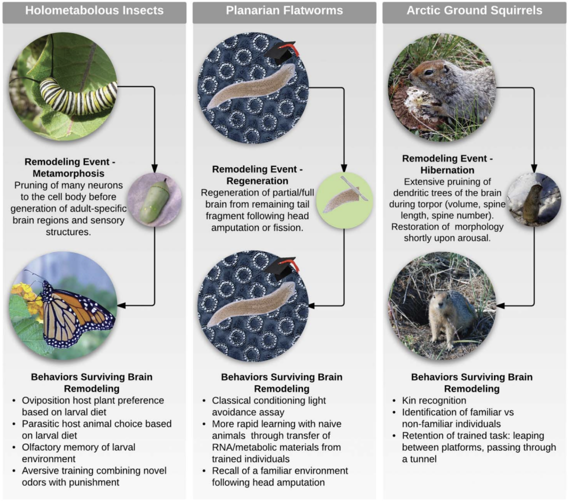
Beyond the synapse

- More radical proposals: neurons may learn and remember more complex forms of information using cell-intrinsic mechanisms that go beyond changes in excitability.
- For example, modification of polynucleotide sequences like RNA could be a learning mechanism, consistent with (controversial!) evidence that memories can be transferred between organisms via RNA.

Beyond the synapse

- More radical proposals: neurons may learn and remember more complex forms of information using cell-intrinsic mechanisms that go beyond changes in excitability.
- For example, modification of polynucleotide sequences like RNA could be a learning mechanism, consistent with (controversial!) evidence that memories can be transferred between organisms via RNA.
- This hypothesis might also explain why memories in some species can survive dramatic synaptic remodeling, such as during metamorphosis, hibernation, and even decapitation.

Beyond the synapse



Holometabolous Insects

Remodeling Event - Metamorphosis
Pruning of many neurons to the cell body before generation of adult-specific brain regions and sensory structures.

Behaviors Surviving Brain Remodeling
- Oviposition host plant preference based on larval diet
- Parasitic host animal choice based on larval diet
- Olfactory memory of larval environment
- Aversive training combining novel odors with punishment

Planarian Flatworms

Remodeling Event - Regeneration
Regeneration of partial/full brain from remaining tail fragment following head amputation or fission.

Behaviors Surviving Brain Remodeling
- Classical conditioning light avoidance assay
- More rapid learning with naive animals through transfer of RNA/metabolic materials from trained individuals
- Recall of a familiar environment following head amputation

Arctic Ground Squirrels

Remodeling Event - Hibernation
Extensive pruining of dendritic trees of the brain during torpor (volume, spine length, spine number). Restoration of morphology shortly upon arousal.

Behaviors Surviving Brain Remodeling
- Kin recognition
- Identification of familiar vs non-familiar individuals
- Retention of trained task: leaping between platforms, passing through a tunnel

[Blackiston & Levin 2015]

Study question

Why do findings of memory transfer and persistence following brain damage challenge synaptic learning models? How might such findings be reconciled with synaptic learning models?

Summary

- Synaptic plasticity can be viewed as approximate gradient descent.

Summary

- Synaptic plasticity can be viewed as approximate gradient descent.
- Several possible biological implementations, such as random feedback or dendritic segregation.

Summary

- Synaptic plasticity can be viewed as approximate gradient descent.
- Several possible biological implementations, such as random feedback or dendritic segregation.
- Learning also happens non-synaptically.