## Backprop for recurrent networks

Sebastian Seung

#### Other flavors of backprop

- Recurrent backprop
  - for training steady states of a recurrent network
- Backprop-through-time
  - for training trajectories of a recurrent network

# Steady state of a recurrent network

Recurrent network

$$x_i = f\left(\sum_j W_{ij} x_j + b_i\right)$$

- feedforward network is a special case

 Goal: maximize some function of the activity vector

$$\max_{W,\mathbf{b}} R(\mathbf{x})$$

#### Implicit function theorem

- x is an implicit function of W and b
- defined as the solution of  $\mathbf{F}(\mathbf{x}, W, \mathbf{b}) = 0$
- where  $\mathbf{F}(\mathbf{x}, W, \mathbf{b}) \equiv \mathbf{x} \mathbf{f}(W\mathbf{x} + \mathbf{b})$
- assuming nonsingular Jacobian  $\partial F_i/\partial x_j$
- This is a local definition.

#### **Recurrent backpropagation**

- Find steady state  $\mathbf{x} = \mathbf{f}(W\mathbf{x} + \mathbf{b})$
- Calculate slopes  $D = diag\{\mathbf{f}'(W\mathbf{x} + \mathbf{b})\}$
- Solve for sensitivity  $(D^{-1} W^T)\hat{\mathbf{u}} = \frac{\partial R}{\partial \mathbf{x}}$
- Weight update  $\Delta W = \eta \hat{\mathbf{u}} \mathbf{x}^T$

### Sensitivity lemma

$$\frac{\partial R}{\partial W_{ij}} = \frac{\partial R}{\partial b_i} x_j$$

- It is sufficient to calculate the derivative with respect to the biases.
- Recurrent backprop is a way of calculating the sensitivities

$$\frac{\partial R}{\partial b_i} \equiv \hat{u}_i$$

#### Input as a function of output

What input b is required to make x a steady state?

$$b_i = f^{-1}(x_i) - \sum_j W_{ij} x_j$$

• This is unique, even when output is not a unique function of the input!

#### Jacobian matrix

$$b_{i} = f^{-1}(x_{i}) - \sum_{j} W_{ij} x_{j}$$
$$\frac{\partial b_{i}}{\partial x_{j}} = f^{-1}(x_{i}) \delta_{ij} - W_{ij}$$
$$= \left(D^{-1} - W\right)_{ij}$$

#### **Composition of functions**

 This composition of functions may seem more natural

$$\mathbf{b} \to \mathbf{x} \to R$$

But this composition can also be defined

$$\mathbf{x} \to \mathbf{b} \to R$$

• Both definitions are locally valid.

## Chain rule $\mathbf{x} \rightarrow \mathbf{b} \rightarrow R$

$$\frac{\partial R}{\partial x_j} = \sum_i \frac{\partial R}{\partial b_i} \frac{\partial b_i}{\partial x_j}$$
$$= \sum_i \frac{\partial R}{\partial b_i} \left( D^{-1} - W \right)_{ij}$$
$$\frac{\partial R}{\partial \mathbf{x}} = \left( D^{-1} - W^T \right) \frac{\partial R}{\partial \mathbf{b}}$$

#### **Trajectory learning**

- Initialize at  $\mathbf{x}(0)$ , iterate for T time steps  $x_i(t) = f\left(\sum_j W_{ij} x_j(t-1) + b_i\right)$
- Goal: maximize some function of the time series of activity vectors

$$\max_{W,\mathbf{b}} R(\mathbf{x}(1),\ldots,\mathbf{x}(T))$$

#### Backpropagation through time

- Multilayer perceptron
  - Same number of neurons in each layer
  - Same weights and biases in each layer (weight-sharing)

$$\mathbf{x}(0) \xrightarrow{W,\mathbf{b}} \mathbf{x}(1) \xrightarrow{W,\mathbf{b}} \cdots \xrightarrow{W,\mathbf{b}} \mathbf{x}(T)$$
$$\hat{\mathbf{u}}(1) \xleftarrow{W^{T}} \hat{\mathbf{u}}(2) \xleftarrow{W^{T}} \cdots \xleftarrow{W^{T}} \hat{\mathbf{u}}(T+1)$$

#### Forward pass

• Initial condition **x**(0)

$$\mathbf{u}(t) = W\mathbf{x}(t-1) + \mathbf{b}(t)$$
$$\mathbf{x}(t) = \mathbf{f}(\mathbf{u}(t))$$

#### **Backward pass**

• Final condition  $\hat{\mathbf{u}}(T+1) = 0$ 

$$\hat{\mathbf{x}}(t) = W^T \hat{\mathbf{u}}(t+1) + \frac{\partial R}{\partial \mathbf{x}(t)}$$

$$\hat{\mathbf{u}}(t) = D(t)\hat{\mathbf{x}}(t)$$

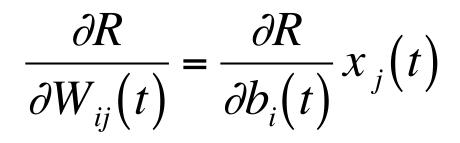
#### Weight update

$$\Delta W = \eta \sum_{t} \hat{\mathbf{u}}(t) \mathbf{x}(t-1)^{T} \quad \Delta b = \eta \sum_{t} \hat{\mathbf{u}}(t)$$

#### Sensitivity lemma

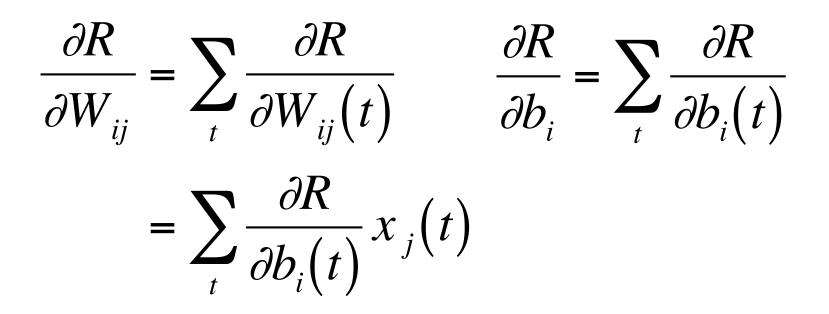
 Suppose that weights and biases are functions of time:

$$x_i(t) = f\left(\sum_j W_{ij}(t)x_j(t-1) + b_i(t)\right)$$



#### Sensitivity lemma

 If the weights and biases are constant in time,



#### Input as a function of output

$$\mathbf{x}(t) = f(W\mathbf{x}(t-1) + \mathbf{b}(t))$$
$$\mathbf{b}(t) = f^{-1}(\mathbf{x}(t)) - W\mathbf{x}(t-1)$$

#### Jacobian matrix

$$\mathbf{b}(t) = \mathbf{f}^{-1}(\mathbf{x}(t)) - W\mathbf{x}(t-1)$$
$$\frac{\partial b_i(t)}{\partial x_j(t')} = \delta_{tt'}(D^{-1}(t))_{ij} - W_{ij}\delta_{t-1,t'}$$
$$D(t) = \operatorname{diag}\left\{\mathbf{f}'(W\mathbf{x}(t-1) + \mathbf{b}(t))\right\}$$

#### Chain rule

$$\frac{\partial R}{\partial x_j(t')} = \sum_{i,t} \frac{\partial R}{\partial b_i(t)} \frac{\partial b_i(t)}{\partial x_j(t')}$$
$$= \sum_i \hat{u}_i(t') (D^{-1}(t'))_{ij} - \sum_i \hat{u}_i(t'+1) W_{ij}$$
$$\frac{\partial R}{\partial \mathbf{x}(t)} = D^{-1}(t) \hat{\mathbf{u}}(t) - W^T \hat{\mathbf{u}}(t+1)$$

#### Lagrangian method

- Application of the chain rule can be confusing.
- Lagrange multipliers provide
  - a "turn the crank" method of calculating gradients
  - another interpretation of the backward pass



Lagrange multiplier

$$L(\mathbf{x}, \hat{\mathbf{u}}, W, \mathbf{b}) = R(\mathbf{x}) - \widehat{\mathbf{u}^{T}} [\mathbf{f}^{-1}(\mathbf{x}) - W\mathbf{x} - \mathbf{b}]$$

steady state constraint

#### Stationary point

 $\mathbf{x}^*(W, \mathbf{b})$  and  $\hat{\mathbf{u}}^*(W, \mathbf{b})$  such that

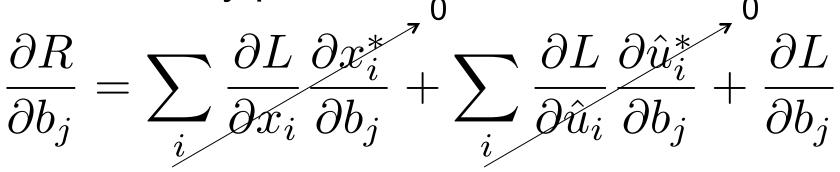
$$0 = -\frac{\partial L}{\partial \hat{\mathbf{u}}} = \mathbf{f}^{-1}(\mathbf{x}) - W\mathbf{x} - \mathbf{b}$$
$$0 = \frac{\partial L}{\partial \mathbf{x}} = \frac{\partial R}{\partial \mathbf{x}} - (D^{-1} - W^T)\hat{\mathbf{u}}$$

# The Lagrangian equals the reward function at a stationary point

 $R(\mathbf{x}^*(W, \mathbf{b})) = L(\mathbf{x}^*(W, \mathbf{b}), \hat{\mathbf{u}}^*(W, \mathbf{b}), W, \mathbf{b})$ 

# Sensitivities of the Lagrangian and reward function

• All derivatives are evaluated at the stationary point.



• Similarly  $\frac{\partial R}{\partial W_{ij}} = \frac{\partial L}{\partial W_{ij}}$ 

## The reward gradients

$$\frac{\partial}{\partial W} R(\mathbf{x}^*(W, \mathbf{b})) = \frac{\partial L}{\partial W} = \hat{\mathbf{u}}\mathbf{x}^T$$
$$\frac{\partial}{\partial \mathbf{b}} R(\mathbf{x}^*(W, \mathbf{b})) = \frac{\partial L}{\partial \mathbf{b}} = \hat{\mathbf{u}}$$

where it's understood that

 $\hat{\mathbf{u}} = \hat{\mathbf{u}}^*(W, \mathbf{b}) \qquad \mathbf{x} = \mathbf{x}^*(W, \mathbf{b})$