Meta-Learning Backpropagation & Improving It

Variable Shared Meta Learning (VS-ML)

**Background: General Meta Learning**
- **General Meta Learning** seeks to automate learning algorithm design
- Learning Algorithms (LAs) should be reusable across a wide range of environments/tasks
- **Current Meta Learning** either doesn’t generalize very well or assumes many human engineered components (e.g. backpropagation & loss functions)

**VS-ML**
- $|V_L| \gg |V_M|$ Meta variables that encode the learning algorithm
- $|V_L|$ Learned variables that encode what is being learned

**Variable Shared Meta Learning**
- Very simple: Take a ‘neural network’ and add variable sharing and sparsity
- Meta-learned LAs generalize due to many more $|V_L|$ (here activations) than $|V_M|$ (here parameters of the network)
- Unifies Learned Learning Rules / Fast Weights & Meta RNNs
- Everything is a variable: variables we thought of as activations can be interpreted as weights, LSTM defines time-scales

**Variable Shared Meta RNN**
- Multiple RNNs with shared parameters $V_M$
- RNNs are connected (e.g. like weights in a neural network)
- Can implement backpropagation and other learning algorithms

**Neural interpretation**
- Can be interpreted as more complex weights and neurons

**VS-ML generalizes**
- **Learned Learning Rules & Fast weights**
  - Interpreting the weight as just another activation, everything is a variable
  - All activations are learned variables
  - Becomes a **generic RNN** (LSTM)
  - Now the forward function is also learned

**Generalize update rules**
- Interpreting the weight as just another activation, everything is a variable
- All activations are learned variables
- Becomes a **generic RNN** (LSTM)
- Now the forward function is also learned

**Duplicate RNNs & Choose interactions**
- Can be interpreted as more complex weights and neurons

**Experiments**
- **Meta Learning Backpropagation**
  - Can an LSTM implement backpropagation purely in its recurrent dynamics? Yes!
  - A sort of ‘Learning Algorithm Cloning’

**Interpretation of VS-ML as a Meta RNN**
- **Simple model** - just a vanilla RNN / LSTM
- **Meta Variables** $V_M$: Parameters of a RNN
- **Learned Variables** $V_L$: State / Activations of an RNN
- **Crucial ingredient:** Feedback signal (e.g. reward) needs to be fed as an input [Hochreiter et al 2001]

**Experiments**
- **Meta Learning Supervised Learning from Scratch**
  - Learning is completely online, no batching, no pre-training
  - Meta Testing on MNIST (within distribution)
  - Learn on MNIST, test on Fashion MNIST (out of distribution)

**Sample efficient**
- Use LSTM and run on Fashion MNIST (out of distribution)

**Strong generalization**
- No gradients required during meta-testing