# Module 8.11 : Dropout

## Other forms of regularization

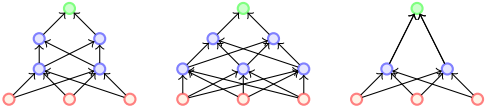
- $L_2$ regularization
- Dataset augmentation
- Parameter Sharing and tying
- Adding Noise to the inputs
- Adding Noise to the outputs
- Early stopping
- Ensemble methods
- Dropout

## Other forms of regularization

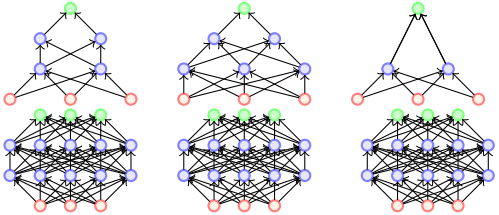
- $L_2$ regularization
- Dataset augmentation
- Parameter Sharing and tying
- Adding Noise to the inputs
- Adding Noise to the outputs
- Early stopping
- Ensemble methods
- Dropout

- Typically model averaging(bagging ensemble) always helps

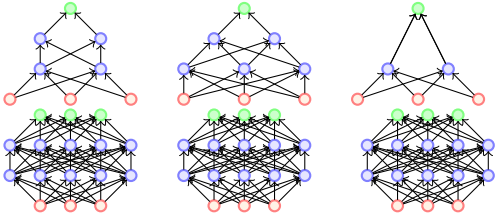
- Typically model averaging(bagging ensemble) always helps
- Training several large neural networks for making an ensemble is prohibitively expensive



- Typically model averaging(bagging ensemble) always helps
- Training several large neural networks for making an ensemble is prohibitively expensive
- Option 1: Train several neural networks having different architectures(obviously expensive)

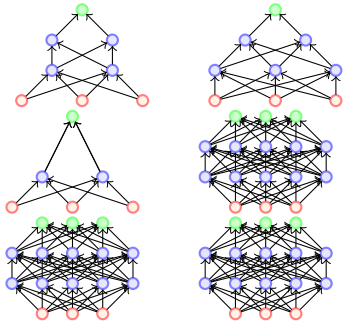


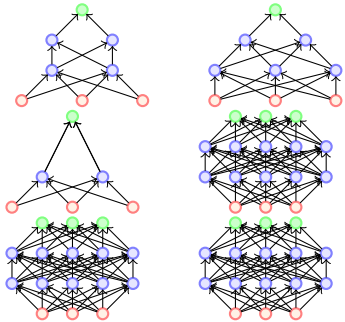
- Typically model averaging(bagging ensemble) always helps
- Training several large neural networks for making an ensemble is prohibitively expensive
- Option 1: Train several neural networks having different architectures(obviously expensive)
- Option 2: Train multiple instances of the same network using different training samples (again expensive)



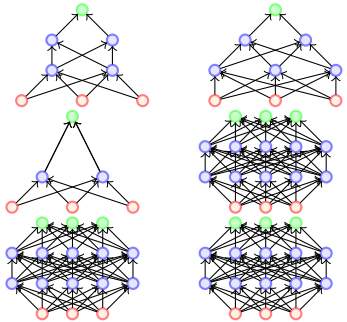
- Typically model averaging(bagging ensemble) always helps
- Training several large neural networks for making an ensemble is prohibitively expensive
- Option 1: Train several neural networks having different architectures(obviously expensive)
- Option 2: Train multiple instances of the same network using different training samples (again expensive)
- Even if we manage to train with option 1 or option 2, combining several models at test time is infeasible in real time applications

- Dropout is a technique which addresses both these issues.

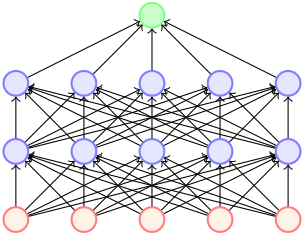




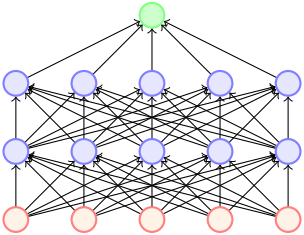
- Dropout is a technique which addresses both these issues.
- Effectively it allows training several neural networks without any significant computational overhead.

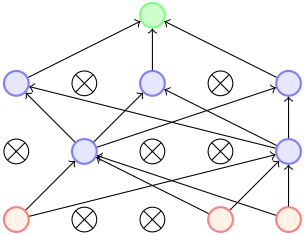


- Dropout is a technique which addresses both these issues.
- Effectively it allows training several neural networks without any significant computational overhead.
- Also gives an efficient approximate way of combining exponentially many different neural networks.

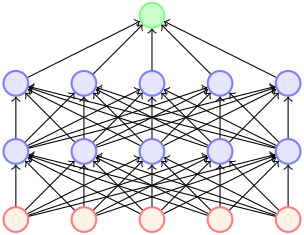


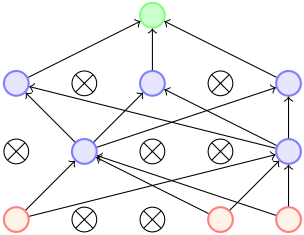
- Dropout refers to dropping out units



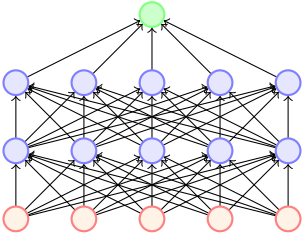


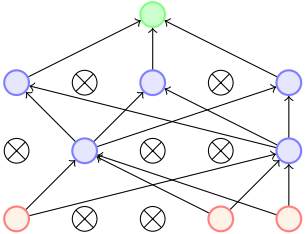
- Dropout refers to dropping out units
- Temporarily remove a node and all its incoming/outgoing connections resulting in a thinned network

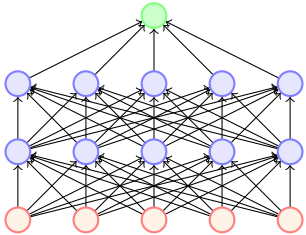


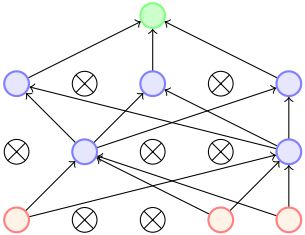


- Dropout refers to dropping out units
- Temporarily remove a node and all its incoming/outgoing connections resulting in a thinned network
- Each node is retained with a fixed probability (typically $p = 0.5$) for hidden nodes and $p = 0.8$ for visible nodes

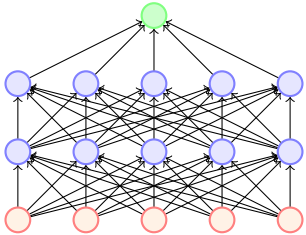


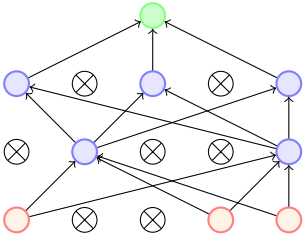




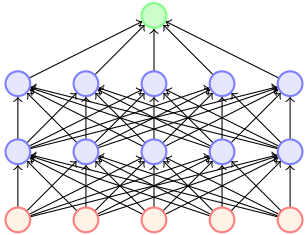


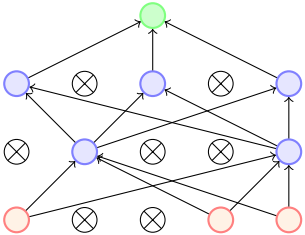
- A neural network with $n$ nodes can be seen as a collection of $2^n$ possible thinned networks



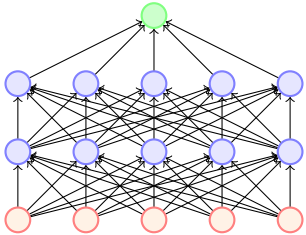


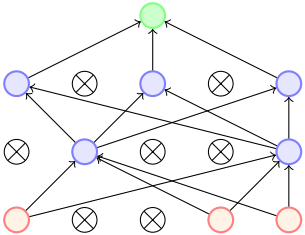
- A neural network with $n$ nodes can be seen as a collection of $2^n$ possible thinned networks
- The weights in these networks are shared



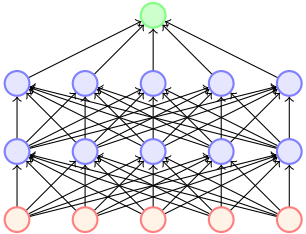


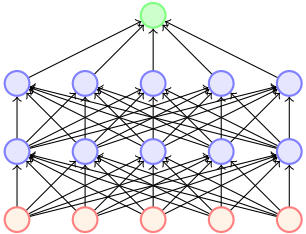
- A neural network with $n$ nodes can be seen as a collection of $2^n$ possible thinned networks
- The weights in these networks are shared
- For each training instance, a different thinned network is sampled and trained

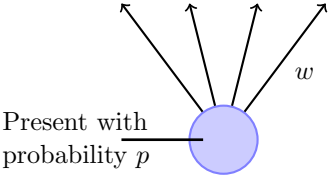




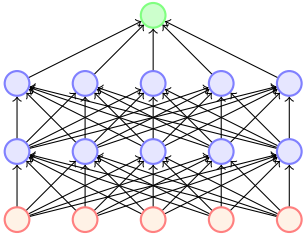
- A neural network with $n$ nodes can be seen as a collection of $2^n$ possible thinned networks
- The weights in these networks are shared
- For each training instance, a different thinned network is sampled and trained
- Each thinned network gets trained rarely (or even never) but the parameter sharing ensures that no model has untrained or poorly trained parameters

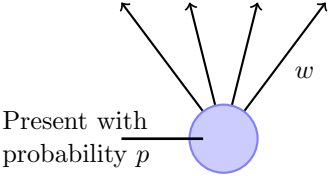




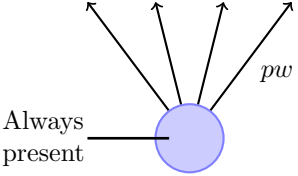



- What happens at test time?



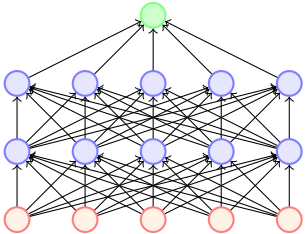


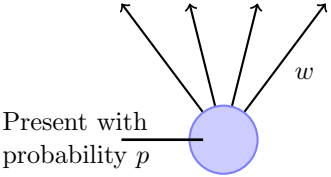

At training time



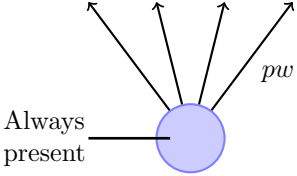

At test time

- What happens at test time?
- Impossible to aggregate the outputs of $2^n$ thinned networks.



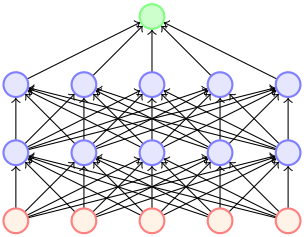


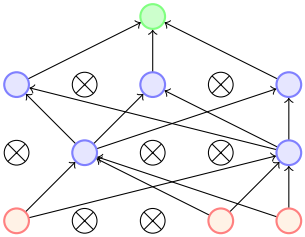

At training time

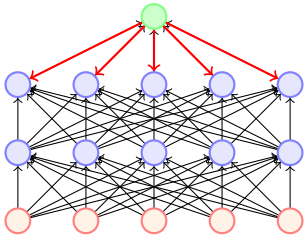


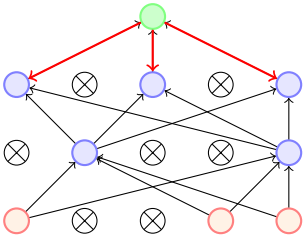

At test time

- What happens at test time?
- Impossible to aggregate the outputs of $2^n$ thinned networks.
- Instead we use the full Neural Network and scale the output of each node by the fraction of times it was on during training.

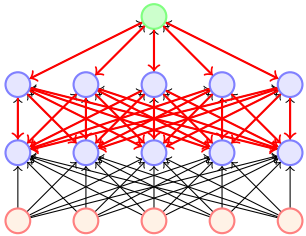


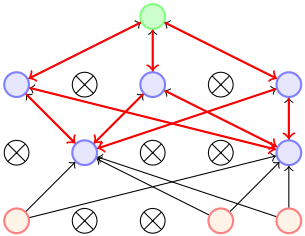




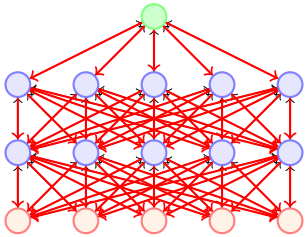


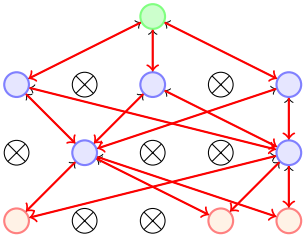
- How do you do backpropagation in such a noisy network which changes for each training instance (or batch)



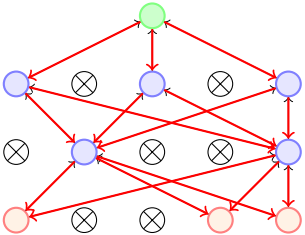


- How do you do backpropagation in such a noisy network which changes for each training instance (or batch)
- Simple: we only backpropagate over the paths which are active and only update those weights which are active in the current thinned network

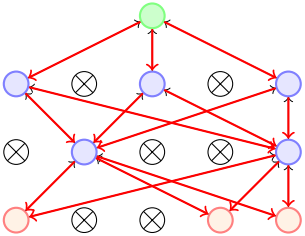




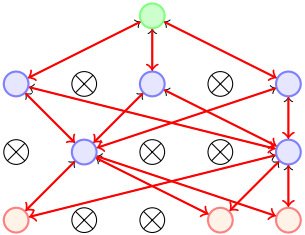
- How do you do backpropagation in such a noisy network which changes for each training instance (or batch)
- Simple: we only backpropagate over the paths which are active and only update those weights which are active in the current thinned network



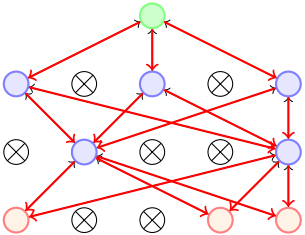
- Dropout essentially applies a masking noise to the hidden units



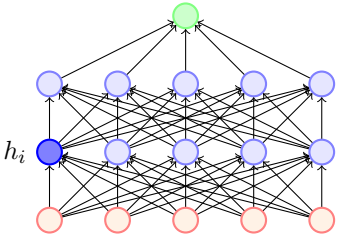
- Dropout essentially applies a masking noise to the hidden units
- Prevents hidden units from co-adapting

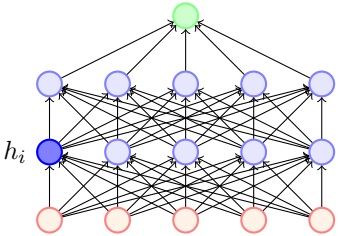


- Dropout essentially applies a masking noise to the hidden units
- Prevents hidden units from co-adapting
- Essentially a hidden unit cannot rely too much on other units as they may get dropped out any time

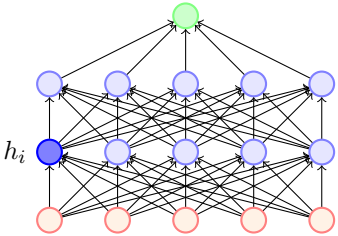


- Dropout essentially applies a masking noise to the hidden units
- Prevents hidden units from co-adapting
- Essentially a hidden unit cannot rely too much on other units as they may get dropped out any time
- Each hidden unit has to learn to be more robust to these random dropouts

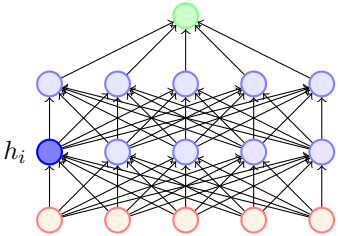

$h_i$



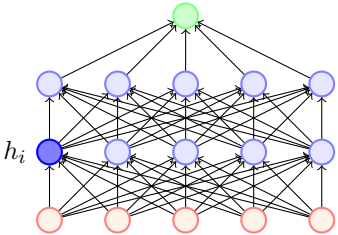

- Here is an example of how dropout helps in ensuring redundancy and robustness



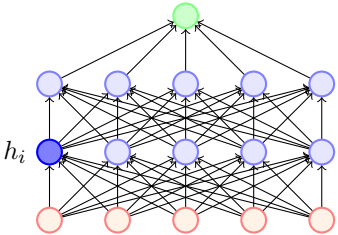

- Here is an example of how dropout helps in ensuring redundancy and robustness
- Suppose $h_i$ learns to detect a face by firing on detecting a nose




- Here is an example of how dropout helps in ensuring redundancy and robustness
- Suppose $h_i$ learns to detect a face by firing on detecting a nose
- Dropping $h_i$ then corresponds to erasing the information that a nose exists




- Here is an example of how dropout helps in ensuring redundancy and robustness
- Suppose $h_i$ learns to detect a face by firing on detecting a nose
- Dropping $h_i$ then corresponds to erasing the information that a nose exists
- The model should then learn another $h_i$ which redundantly encodes the presence of a nose




- Here is an example of how dropout helps in ensuring redundancy and robustness
- Suppose $h_i$ learns to detect a face by firing on detecting a nose
- Dropping $h_i$ then corresponds to erasing the information that a nose exists
- The model should then learn another $h_i$ which redundantly encodes the presence of a nose
- Or the model should learn to detect the face using other features

## Recap

- $L_2$ regularization
- Dataset augmentation
- Parameter Sharing and tying
- Adding Noise to the inputs
- Adding Noise to the outputs
- Early stopping
- Ensemble methods
- Dropout