Neural Networks in Practice

Many ways to improve weight learning in NNs

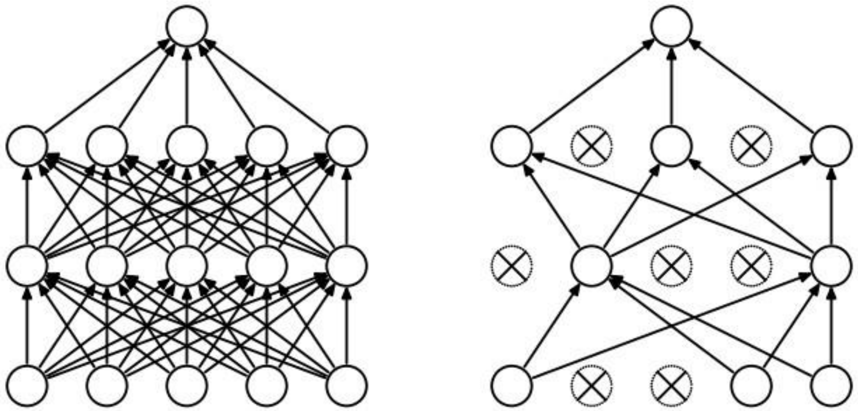
- Use **regularized squared loss (cost)** prediction (can still use backpropagation in this setting)

$$C(y_{\text{true}}, y_{\text{pred}}) = \frac{1}{2}(y - f(\boldsymbol{x}; \boldsymbol{w}, b))^2 + \frac{\lambda}{2}\|w\|_2^2$$

    - L_1 regularization can also be useful
    - $\lambda > 0$ should be chosen with a **validation set**
- Try other loss functions, e.g., the **cross entropy**
    - $C(y_{\text{true}}, y_{\text{pred}}) - y \log f(\boldsymbol{x}) - (1 - y) \log(1 - f(x))$
- **Initialize weights** of the network more cleverly
    - Random initializations are likely to be far from optimal
- Learning procedure can have **numerical difficulties** if there are a **large number of layers**
    - **Early stopping**: stop the learning early in the hopes that this prevents overfitting

Drop out: A **heuristic** **bagging-style approach** applied to neural networks to **counteract overfitting**

- **Randomly remove** a certain percentage of **neurons from the network** and then train only on the remaining neurons
- networks **recombined using an approximate averaging**
- keeping around too many networks and doing proper bagging can be costly in practice



(a) Standard Neural Net

(b) After applying dropout.

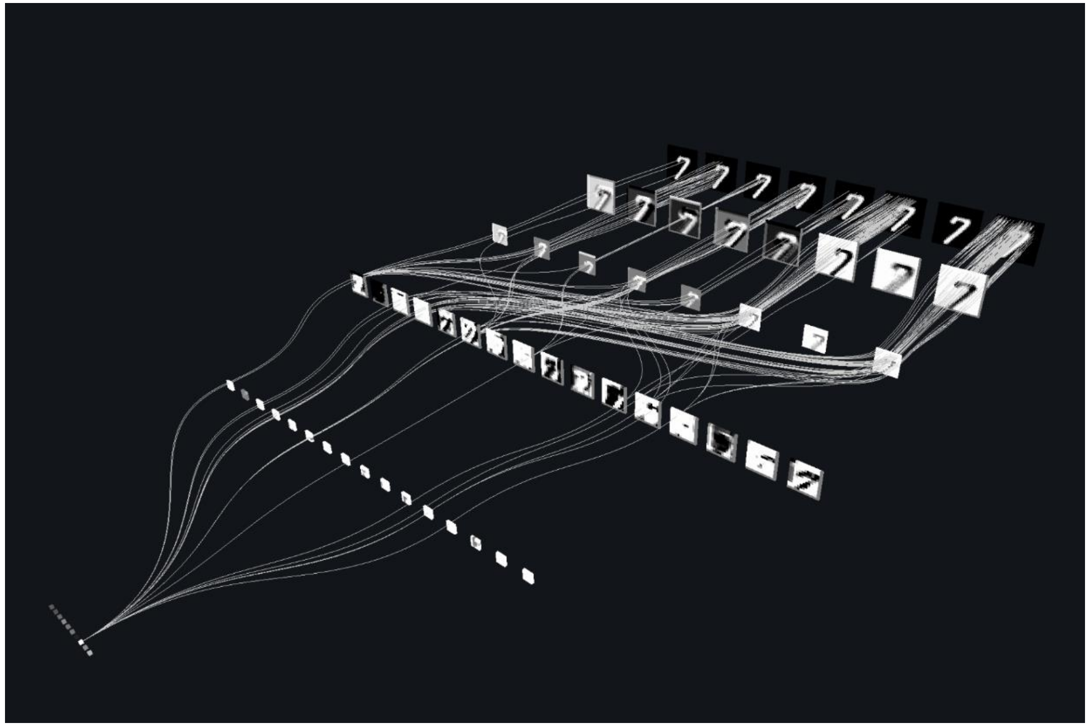
Parameter Tying

Parameter tying: Assume some of the weights in the model are the same to reduce the **dimensionality** of the learning problem;

- Also a way to learn "simpler" models
- Can lead to significant compression in neural networks (i.e., >90%)

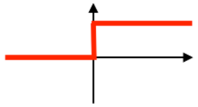
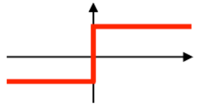
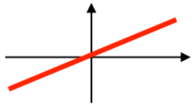
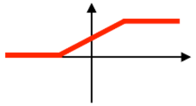
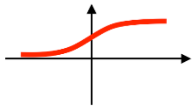
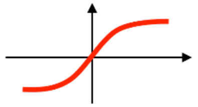
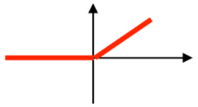
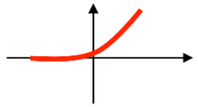
Convolutional neural networks

- Instead of the output of every neuron at layer ℓ being used as an input to every neuron at layer $\ell + 1$, edges between layers are chosen more locally
- Many tied weights and biases
  - convolution nets apply the same process to many different local chunks of neurons
- Often combined with pooling layers
  - layers that replacing small regions of neurons with their aggregated output
- Used extensively for image classification tasks



Topological Visualization of a Convolutional Neural Network by Terence Broad http://terencebroad.com/nnvis.html

Activation Functions

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Example: Self Driving Cars