RLearning: Short guides to reinforcement learning

Unit 4-2: Deep Neural Networks

Davud Rostam-Afschar (Uni Mannheim)

How to improve flexibility of approximation?

Deep Neural Networks

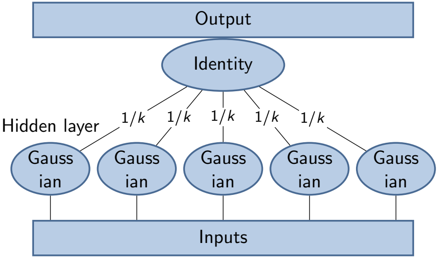
Deep Neural Networks

- Definition: neural network with many hidden layers
- Advantage: high expressivity
- Challenges:
  - How should we train a deep neural network?
  - How can we avoid overfitting?

(Goodfellow, Bengio, and Courville, 2016)

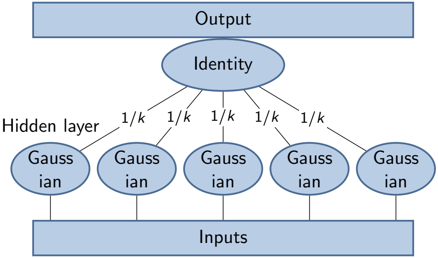
Mixture of Gaussians

- Shallow neural network (flat mixture)

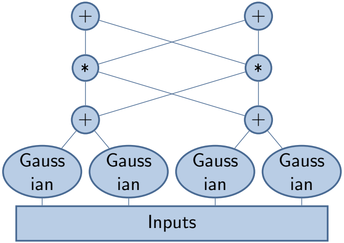


Mixture of Gaussians

- Shallow neural network (flat mixture)

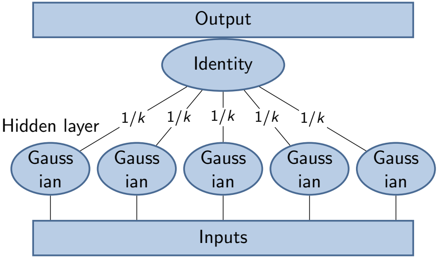


- Deep neural network (hierarchical mixture)

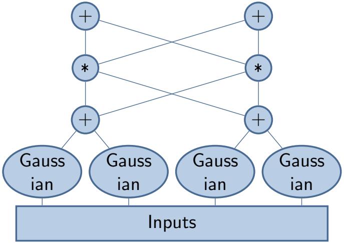


Mixture of Gaussians

- Shallow neural network (flat mixture)



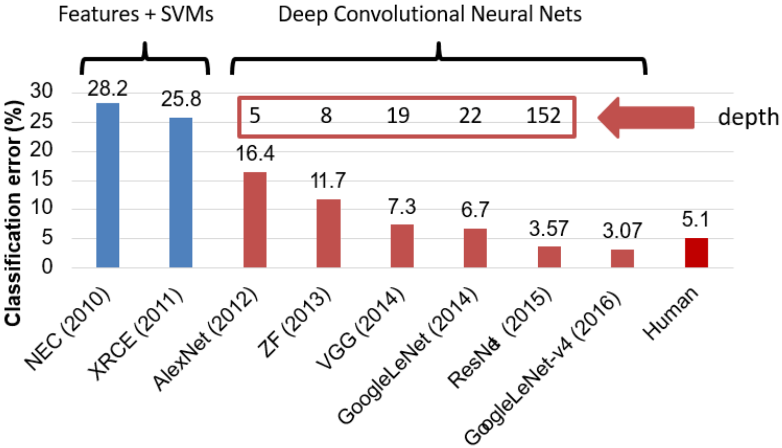
- Deep neural network (hierarchical mixture)



Sum-Product Network (Exponentially large mixture of Gaussians but linear hierachy)

Image Classification

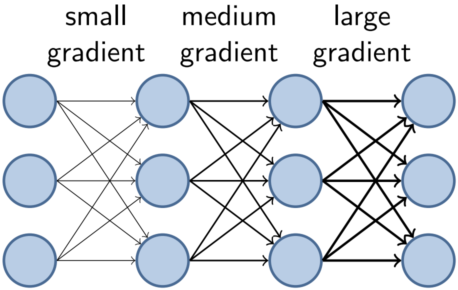
- ImageNet Large Scale Visual Recognition Challenge



Vanishing Gradients

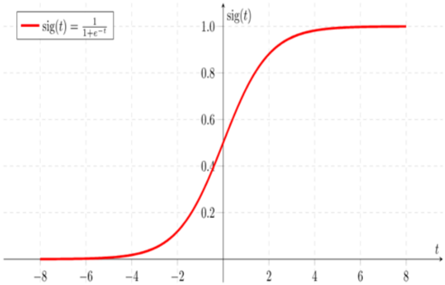
Vanishing Gradients

- Deep neural networks of sigmoid and hyperbolic units often suffer from vanishing gradients

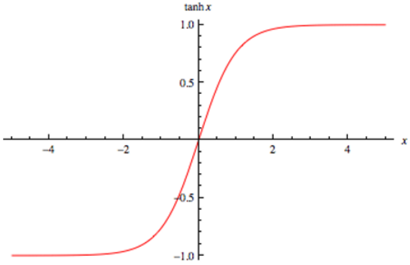


Sigmoid and hyperbolic units

- Derivative is always less than 1



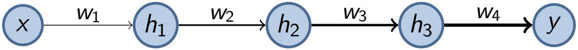
sigmoid



hyperbolic

Simple Example

- $y = \sigma\left(w_4\sigma\left(w_3\sigma\left(w_2\sigma\left(w_1 x\right)\right)\right)\right)$



- Common weight initialization in (-1,1)
- Sigmoid function and its derivative always less than 1
- This leads to vanishing gradients:

$$\frac{\partial y}{\partial w_4} = \sigma'(a_4)\,\sigma(a_3)$$
$$\frac{\partial y}{\partial w_3} = \sigma'(a_4)\,w_4\sigma'(a_3)\,\sigma(a_2) \leq \frac{\partial y}{\partial w_4}$$
$$\frac{\partial y}{\partial w_2} = \sigma'(a_4)\,w_4\sigma'(a_3)\,w_3\sigma'(a_2)\,\sigma(a_1) \leq \frac{\partial y}{\partial w_3}$$
$$\frac{\partial y}{\partial w_1} = \sigma'(a_4)\,w_4\sigma'(a_3)\,w_3\sigma'(a_2)\,w_2\sigma'(a_1)\,x \leq \frac{\partial y}{\partial w_2}$$

Mitigating Vanishing Gradients

- Some popular solutions:
  - Pre-training
  - **Rectified linear units**
  - Batch normalization
  - Skip connections

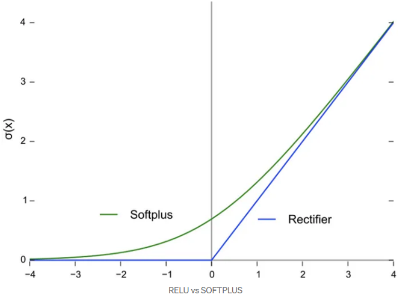
Rectified Linear Units

- Rectified linear: $h(a) = \max(0, a)$
  - Gradient is 0 or 1
  - Sparse computation

Rectified Linear Units

- Rectified linear: $h(a) = \max(0, a)$
  - Gradient is 0 or 1
  - Sparse computation

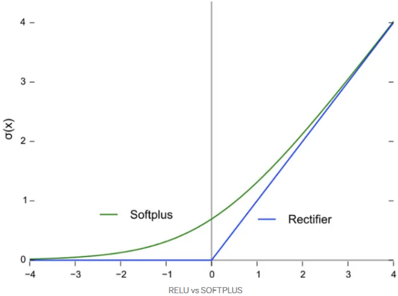
- Soft version ("Softplus"): $h(a) = \log\left(1 + e^{a}\right)$



Rectified Linear Units

- Rectified linear: $h(a) = \max(0, a)$
  - Gradient is 0 or 1
  - Sparse computation

- Soft version ("Softplus"): $h(a) = \log\left(1 + e^{a}\right)$



- Warning: softplus does not prevent gradient vanishing (gradient < 1)

References I

Goodfellow, I., Y. Bengio, and A. Courville (2016): *Deep learning*, vol. 196. MIT press, Available at http://deeplearningbook.org/.

Takeaways

How do Deep Neural Networks Help Modeling Complex Data?

- Use multiple hidden layers
- They enable complex function approximation
- A key challenge is the vanishing gradient problem
- Solutions include ReLU activation functions, batch normalization, skip connections, and pre-training
- Rectified Linear Units (ReLU) help mitigate vanishing gradients