What is the intuition behind neural networks?

How do neural networks learn?

How to train neural networks?

Machine Learning: 4 Main Components

- Data
- Model/Representation
- Cost/Error/Loss of model
- Model Optimizer
Dealing with Edge Conditions

Dealing with Edge Conditions

Dealing with Edge Conditions
XOR Perceptron

AND

OR

XOR Perceptron

AND

OR

XOR

XOR Perceptron

XOR

XOR
Iris Dataset

How Neural Networks Learn

- Data: Iris dataset
- Model: 3 layer neural network

Activation

Output: \[ y_k = \frac{1}{1 + e^{-x_k}} \]

Hidden: \[ h_j = \frac{1}{1 + e^{-x_j}} \]

- Input layer (Length, Width)
- Hidden layer (nodes)
- Output layer (0, 1, 2)

Restrictions
- setosa
- versicolor
- virginica
How Neural Networks Learn

1. Parameter initialization
2. Data input
3. Forward propagation
4. Loss calculation
5. Backpropagation = updates

\[ \Delta w = \eta \frac{\partial L}{\partial w} = \eta \frac{\partial L}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial w} \]

8/1/2018
How Neural Networks Learn

1. Parameter initialization
2. Data input
3. Forward propagation
4. Loss calculation
5. Backpropagation + updates

Gradient Descent Flavors

- vanilla gradient descent - entire dataset
- stochastic gradient descent - random batch of samples (IID)
- online gradient descent - (need not be IID)
Gradient Descent Flavors

vanilla gradient descent - entire dataset
stochastic gradient descent - random batch of examples (IID)
online gradient descent - (need not be IID)

learning rate  batch size  # of epochs

What is the intuition behind neural networks?

How do neural networks learn?

How to train neural networks?

The Perfect Fit

Image: Perfect Fit
The Perfect Fit

Hyperparameters

Architecture

parameters vs hyperparameters
Activations

- **step**
  - non-differentiable

- **sigmoid**
  - non-linear, smooth transition
  - values range from 0 to 1

- **tanh**
  - hyperbolic tangent
  - symmetric around zero

- **ReLU**
  - Rectified Linear Unit
  - outputs 0 for negative inputs

- **Leaky ReLU**
  - adds a small constant for negative inputs

- **ELU**
  - Exponential Linear Unit

Initializations

- **0** - stuck at a saddle point
- **constants** - difficult to break the symmetry
- **large random values** - small gradients, slow convergence

Initializations
Initializations

<table>
<thead>
<tr>
<th>Name</th>
<th>α</th>
<th>β</th>
<th>γ</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>α = 0</td>
<td>β = 0</td>
<td>γ = 0</td>
<td>used by [12][14]</td>
</tr>
<tr>
<td>Xavier/Herbst uniform</td>
<td>α = 0</td>
<td>β = 0</td>
<td>γ = 0</td>
<td>[15][16]</td>
</tr>
<tr>
<td>Xavier/Herbst normal</td>
<td>α = 0</td>
<td>β = 0</td>
<td>γ = 0</td>
<td>[15][16]</td>
</tr>
<tr>
<td>He</td>
<td>α = 0</td>
<td>β = 1</td>
<td>γ = 0</td>
<td>[17][18]</td>
</tr>
<tr>
<td>Glorot</td>
<td></td>
<td></td>
<td>γ = 0</td>
<td>[19][20]</td>
</tr>
</tbody>
</table>

Note: (β) Weight initializations scheme of the form α = a - 1, 1.1, 6, and 0.01 x 7. Traditionally, these are initialized with expected initial weights in the range of -3 to 3. However, these values (a) similar to an initial initialization for ReLU activations. Glorot and He initializations work well with this simple function.

References:

Loss Functions

regression - mean squared error

multiclass classification - categorical cross entropy

pixel classification - dice/Wasserstein dice coefficient
Optimizers

stochastic gradient descent + momentum

Optimizers

stochastic gradient descent + momentum

adaptive gradient (AdaGrad)

Optimizers

stochastic gradient descent + momentum

adaptive gradient (AdaGrad)

root mean square propagation (RMSProp)
Optimizers

Regularizers

L1, L2 regularization

\[ L_{new} = L + \frac{\lambda}{2} |W| \]

\[ L_{new} = L + \frac{\lambda}{2} W^2 \]

Regularizers

L1, L2 regularization

\[ L_{new} = L + \frac{\lambda}{2} |W| \]

\[ L_{new} = L + \frac{\lambda}{2} W^2 \]
Regularizers

dropout

Figure 1. Dropout, Neural Net Model. Left: A standard neural net with 3 hidden layers. Right: An example of a neural net with 3 hidden layers. By applying dropout to the network on the right, the neural net will be less accurate.

Batch Normalization: A Simple Way to Prevent Neural Networks from Overfitting
Journal of Machine Learning Research - 2003

Regularizers

batch normalization

Figure 2. Batch Normalization, Neural Net Model. Left: A standard neural net with 3 hidden layers. Right: An example of a neural net with 3 hidden layers. By applying batch normalization to the network on the right, the neural net will be more accurate.

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Regularizers

batch normalization

Figure 3. Batch Normalization, Neural Net Model. Left: A standard neural net with 3 hidden layers. Right: An example of a neural net with 3 hidden layers. By applying batch normalization to the network on the right, the neural net will be more accurate.
Optimizer-specific Hyperparameters

learning rate
0.1, 0.01, 0.001, 0.0001,...

Optimizer-specific Hyperparameters

learning rate
0.1, 0.01, 0.001, 0.0001,...

Optimizer-specific Hyperparameters

learning rate
0.1, 0.01, 0.001, 0.0001,...
Babysitting your Network

Debugging through Learning Curves