Artificial Neural Networks (ANNs) Deep Learning +3

How do you implement a dropout in deep neural networks?

I looked at the 2012 paper on dropout by G. Hinton, and while I understand its basic concept, I'm confused about its details.
Supposing that (stochastic) backprop is used for training, my question is:

1. Is a different sub-network sampled for each training input? Or is a single sub-network sampled for the whole training inputs?

2. The entire network is used for the test case. What does it mean that the output of a dropped node (in the training case) is multiplied by the retaining rate *p*?

The link to Hinton's paper is here: Page on toronto.edu

I would really appreciate it if a detailed guide on implementing dropout in a NN is provided.

Answer Request Follow 18 Comment Share Downvote

1 Answer

Ottokar Tilk, PhD student at Tallinn University of Technology
Updated Feb 21, 2016 · Upvoted by Erlend Davidson, Academic physics researcher using ML techniques both for work and personal fun.

Originally Answered: How would you implement drop-out in a (deep) neural network?

Lets take an example where you want to use a dropout coefficient of 0.5 in layer 2 of your network.

During training:
The outputs/activations of layer 2 are multiplied elementwise with a binary mask where the probability of each element of the mask being 1 is 0.5 (zero otherwise):

Without dropout: $y_2 = f(z_2)$
With dropout: $y_2 = f(z_2) \circ m_2$

where $f()$ is the activation function (e.g. tanh or ReLU), $\circ$ is an elementwise multiplication operation, z_2 is the input vector of layer 2, m_2 is the binary dropout mask and y_2 is layer 2 output/activation vector.

During testing/validation:
Inputs z_3 to layer 3 are computed as follows:

Without dropout: $z_3 = W_3 y_2$
With dropout: $z_3 = 0.5 W_3 y_2$

where y_2 is the output/activation vector of layer 2 and 0.5 is the dropout coefficient of layer 2.

During backpropagation:

With dropout you need to multiply the derivative of y_2 wrt. z_2 with the same dropout mask m_2 that you used during forward propagation.

A small code example:

(the rest of it is available here)

```
def forward(x, W1, W2, W3, training=False):
    z1 = np.dot(x, W1)
    y1 = np.tanh(z1)

    z2 = np.dot(y1, W2)
    y2 = np.tanh(z2)
```

Related Questions

- Why do neural networks with dropout perform so well?
- What does a dropout in neural networks mean?
- Are physical implementations of neural networks clearly preferable to GPU's?
- How can I use dropout in deep residual network?
- How is dropout applied to mini-batches in dropout neural networks with stochastic gradient descent?
- Andrew Ng: Where should we not implement the neural networks?
- How do I handle both programming and research in area of Neural networks and deep neural networks?
- How real is the benefit of dropout / dropconnect techniques in feed-forward neural networks?
- How training technique for deep neural network such as CNN, SGD, dropout, etc. solve the vanishing gradient problem?
- What's the optimal way to implement a document classification problem using artificial neural networks?

More Related Questions

Question Stats

18 Public Followers
10,269 Views
Last Asked Oct 18, 2016
1 Merged Question
Edits

```
    else:
        m2 = 0.5
    y2 *= m2

    z3 = np.dot(y2, W3)
    y3 = z3 # linear output

    return y1, y2, y3, m2

def backward(x, y1, y2, y3, m2, t, W1, W2, W3):
    dC_dz3 = dC(y3, t)
    dC_dW3 = np.dot(y2.T, dC_dz3)
    dC_dy2 = np.dot(dC_dz3, W3.T)

    dC_dz2 = dC_dy2 * dtanh(y2) * m2
    dC_dW2 = np.dot(y1.T, dC_dz2)
    dC_dy1 = np.dot(dC_dz2, W2.T)

    dC_dz1 = dC_dy1 * dtanh(y1)
    dC_dW1 = np.dot(x.T, dC_dz1)

    return dC_dW1, dC_dW2, dC_dW3
```