

What excites a neuron ?

The inversion process that we described by Deconvolution and Saliency Maps

- Is input dependent
- The Saliency Map for a single (or summary) location at feature map k of layer l
- Depends on a particular input $\mathbf{x}^{(i)}$ being feed to layer 0

By finding the examples that "most excite" the feature map, we were indirectly able to guess at the feature being recognized by the feature map.

We now demonstrate a more direct approach

- Determine the input value
- That excites (causes large values)
- A single location/neuron (or summary) of feature map k of layer l

By finding the *single input* that most excites a feature map, we may interpret the feature map as attempting to recognize similar inputs.

Gradient Ascent: Inverting a Neural Network

We have already introduced the notion of computing the *sensitivity* of a feature

- At spatial location idx of feature map k of layer l
- To a change in the feature at spatial location idx' feature map k' of layer 0

$$s_{(l),\text{idx},k,(0),\text{idx}',k'} = \frac{\partial \mathbf{y}_{(l),\text{idx},k}}{\partial \mathbf{y}_{(0),\text{idx}',k'}}$$

We used this to define Saliency Maps

- Which indicate how much more "excited" $\mathbf{y}_{(l),\text{idx},k}$ becomes
- When we increase the stimulus at layer 0 : $\mathbf{y}_{(0),\text{idx}',k'}$

We also know that Gradient Descent is used

- To find the optimal value \mathbf{W}^* for the weights \mathbf{W} that parameterize the layers of a Neural Network
- By optimizing (find the minimum) a Loss Function
- Using derivatives of the Loss with respect to the weights

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} L(\hat{\mathbf{y}}, \mathbf{y}; \mathbf{W})$$

What happens if we *combine* these two ideas:

- Find the optimal value for input \mathbf{x}^*
- By optimizing (maximizing) the value $o = \mathbf{y}_{(l),\text{idx},k}$
- Using derivatives of o with respect to \mathbf{x} ?

$$\mathbf{x}^* = \underset{\mathbf{x}=\mathbf{y}_{(0)}}{\operatorname{argmax}} \mathbf{y}_{(l),\text{idx},k}$$

(Remember that the value of $o = \mathbf{y}_{(l),\text{idx},k}$ is a function of input \mathbf{x})

That is:

- We can use Gradient Descent
- To find the value $\mathbf{x}^* = \mathbf{y}_{(0)}$
- That, when used as input to the Neural Network
- Maximizes the value of a particular neuron $\mathbf{y}_{(l),\text{idx},k}$
- Using derivatives

$$\frac{\partial \mathbf{y}_{(l),\text{idx},k}}{\partial \mathbf{y}_{(0),\text{idx}',k'}}$$

We start off by initializing $\mathbf{y}_{(0)}$ to random noise.

- Compute $\mathbf{y}_{(l), \text{idx}, k}$ on the Forward Pass
- Compute $\frac{\partial \mathbf{y}_{(l), \text{idx}, k}}{\partial \mathbf{y}_{(0), \text{idx}', k'}}$ given the current $\mathbf{y}_{(0)}$, on the Backward Pass
- Move $\mathbf{y}_{(0)}$ in the direction of the derivative

After some number of epochs, we obtain an $\mathbf{x}^* = \mathbf{y}_{(0)}$ that maximizes $\mathbf{y}_{(l),\text{idx},k}$.

That is: we find the input \mathbf{x}^* that maximally excites $\mathbf{y}_{(l),\text{idx},k}$.

We can then interpret $\mathbf{y}_{(l),\text{idx},k}$ as looking for the feature

"Is like \mathbf{x}^ "*

Since we are maximizing a value (o) rather than minimizing one (the Loss)

- This method is called *Gradient Ascent*
- By multiplying the objective o by -1 we can trivially turn this into a minimization problem

Extensions

Note that this method is *input independent*

- Other than initializing the "guess" $\mathbf{y}_{(0)}$ to a random value
- It does not depend on any particular input \mathbf{x}

Conclusion

Gradient Ascent is a technique for find the input \mathbf{x}^* that is the "paradigmatic" value for a feature at layer l

It is a simple combination of techniques that we have already learned.

You can do many more interesting things with Gradient Ascent

- What if your initial guess is not random noise ?
- What if you add a constraint on \mathbf{x}^* ?

We will explore these ideas in another lecture.

In [4]: `print("Done")`

Done