Layered Architectures

Advanced Topics in High-Performance Computing

Faisal Qureshi

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Logistic regression and layers

Lets look at how we can specify logistic regression as layers. The ability to specify such models as layers is key to designing neural networks. We will also discuss *backpropagation*.

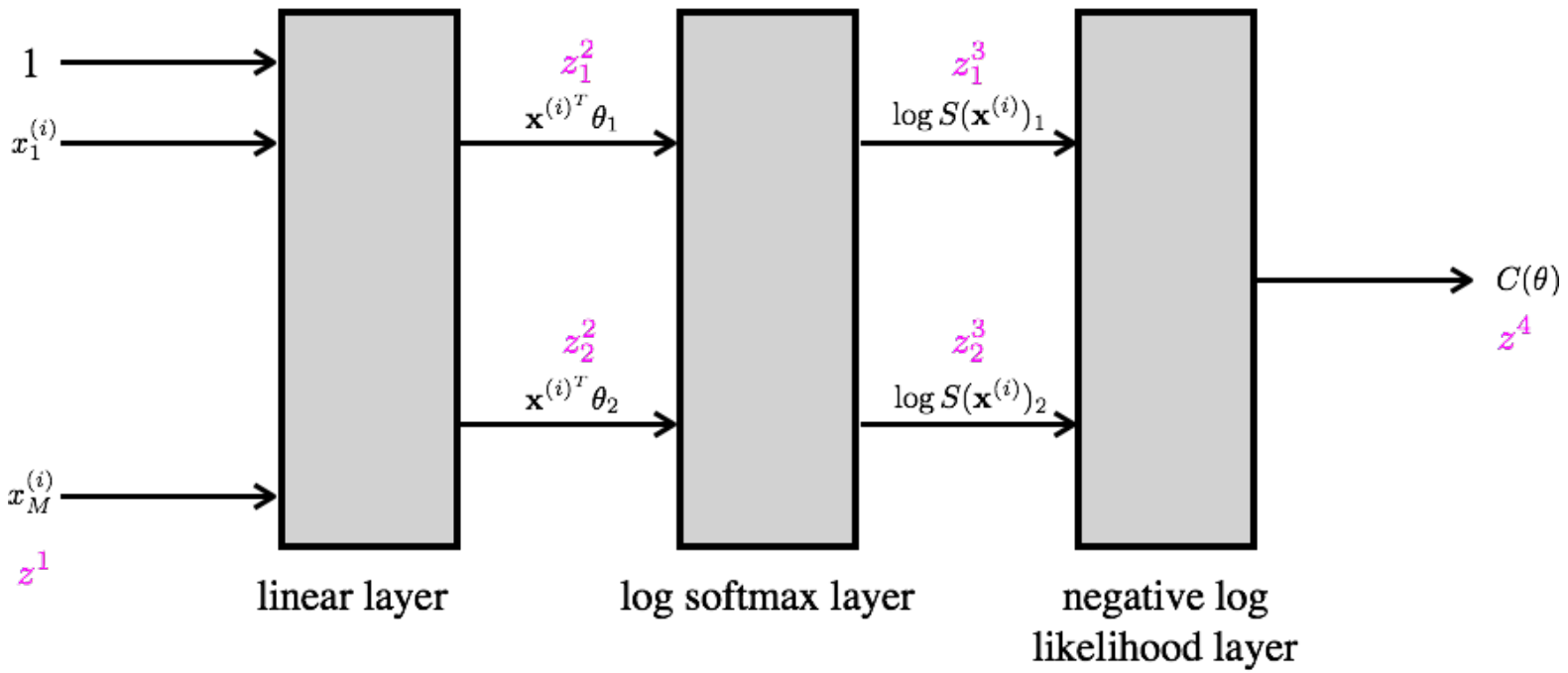
Example: 2-class softmax classifier

Negative log likelihood

$$l(\theta) = -\sum_{i=1}^{N} \mathbb{I}_0(y^{(i)}) \log \frac{e^{\mathbf{x}^{(i)T}\theta_1}}{e^{\mathbf{x}^{(i)T}\theta_1} + e^{\mathbf{x}^{(i)T}\theta_2}} + \mathbb{I}_1(y^{(i)}) \log \frac{e^{\mathbf{x}^{(i)T}\theta_2}}{e^{\mathbf{x}^{(i)T}\theta_1} + e^{\mathbf{x}^{(i)T}\theta_2}}$$

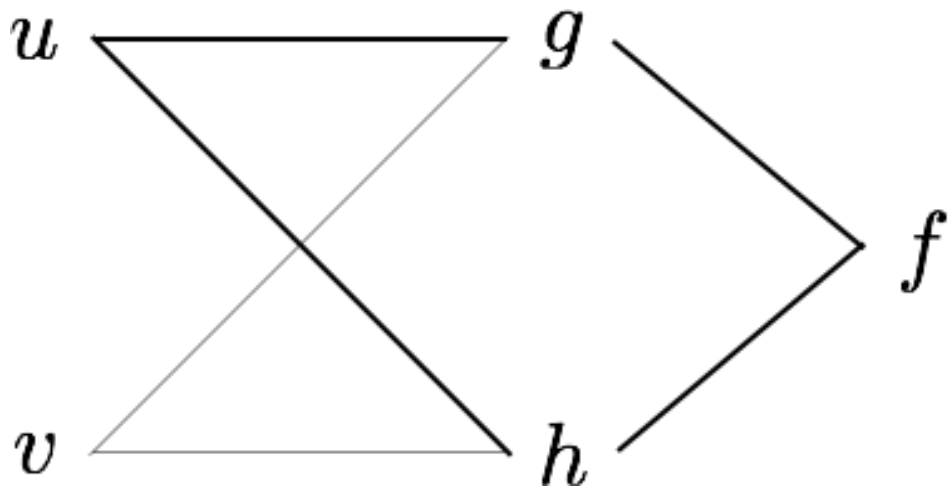
Define cost $C(\theta)$ that we want to minimize to be the negative log likelihood $l(\theta)$.

Layer representation



Chain rule

$$\frac{\partial f(g(u,v),h(u,v))}{\partial u} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial u} + \frac{\partial f}{\partial h}\frac{\partial h}{\partial u}$$



Example: 2-class softmax classifier

We can use the *chain rule* to compute $\frac{\partial z^4}{\partial \theta_1}$ and $\frac{\partial z^4}{\partial \theta_2}$.

$$\frac{\partial z^4}{\partial \theta_1} = \frac{\partial z^4}{\partial z_1^3}\frac{\partial z_1^3}{\partial \theta_1} + \frac{\partial z^4}{\partial z_2^3}\frac{\partial z_2^3}{\partial \theta_1}$$

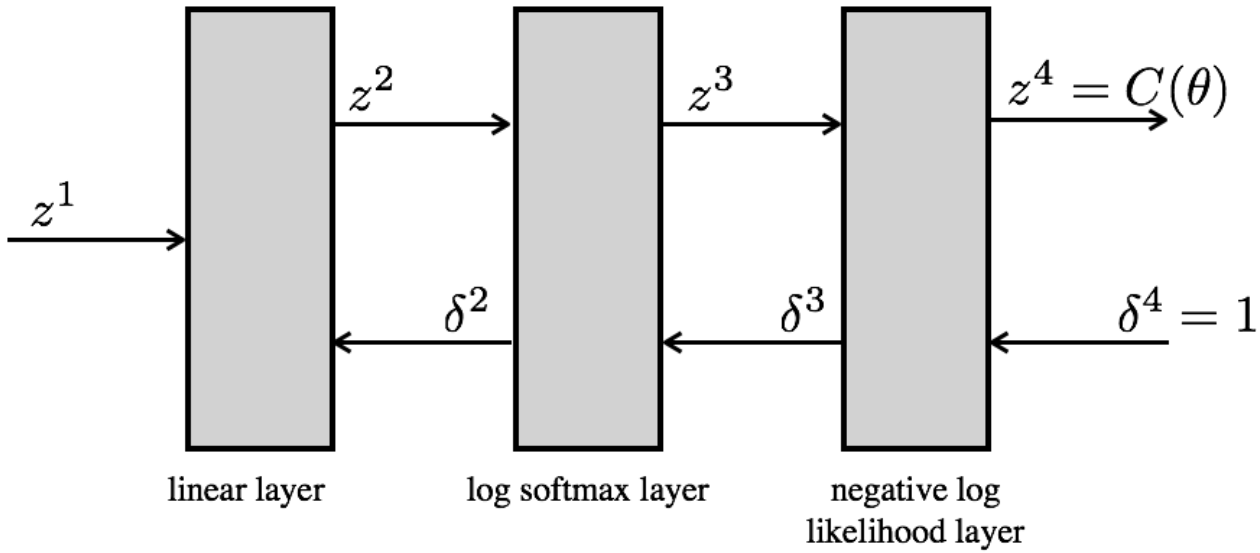
$$= \frac{\partial z^4}{\partial z_1^3}\left(\frac{\partial z_1^3}{\partial z_1^2}\frac{\partial z_1^2}{\partial \theta_1} + \frac{\partial z_1^3}{\partial z_2^2}\frac{\partial z_2^2}{\partial \theta_1}\right)$$

$$+ \frac{\partial z^4}{\partial z_2^3}\left(\frac{\partial z_2^3}{\partial z_1^2}\frac{\partial z_1^2}{\partial \theta_1} + \frac{\partial z_2^3}{\partial z_2^2}\frac{\partial z_2^2}{\partial \theta_1}\right)$$

We can similarly compute $\frac{\partial z^4}{\partial \theta_2}$.

Recall that $z^4 = l(\theta)$, and we can minimize the $l(\theta)$ using gradient descent using the gradients computed above.

Example: 2-class softmax classifier (layered view)



Forward pass

$z^1 = f(\mathbf{x})$ (*input data*)
$z^2 = f(z^1)$ (*linear function*)
$z^3 = f(z^2)$ (*log softmax*)
$z^4 = f(z^3) = l(\theta)$ (*negative log likelihood*, cost)

Backward pass

$\delta^l = \frac{\partial l(\theta)}{\partial z^L}$

Example: 2-class softmax classifier (layered view)

Computing δ^l

$$\delta^4 = \frac{\partial C(\theta)}{\partial z^4} = \frac{\partial z^4}{\partial z^4} = 1$$

$$\delta_1^3 = \frac{\partial C(\theta)}{\partial z_1^3} = \frac{\partial C(\theta)}{\partial z^4}\frac{\partial z^4}{\partial z_1^3} = \delta^4 \frac{\partial z^4}{\partial z_1^3}$$

$$\delta_2^3 = \frac{\partial C(\theta)}{\partial z_2^3} = \frac{\partial C(\theta)}{\partial z^4}\frac{\partial z^4}{\partial z_2^3} = \delta^4 \frac{\partial z^4}{\partial z_2^3}$$

$$\delta_1^2 = \frac{\partial C(\theta)}{\partial z_1^2} = \sum_k \frac{\partial C(\theta)}{\partial z_k^3}\frac{\partial z_k^3}{\partial z_1^2} = \sum_k \delta_k^3 \frac{\partial z_k^3}{\partial z_1^2}$$

$$\delta_2^2 = \frac{\partial C(\theta)}{\partial z_2^2} = \sum_k \frac{\partial C(\theta)}{\partial z_k^3}\frac{\partial z_k^3}{\partial z_2^2} = \sum_k \delta_k^3 \frac{\partial z_k^3}{\partial z_2^2}$$

For any differentiable layer l

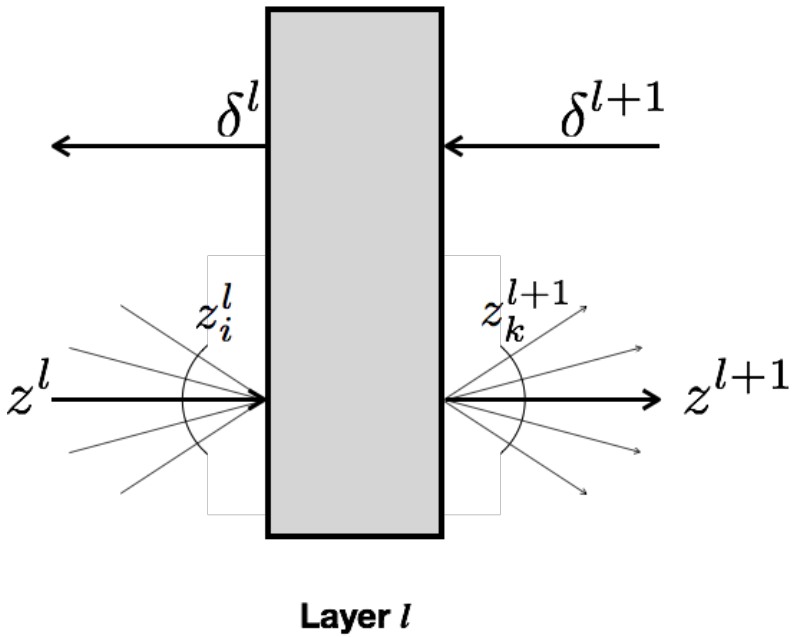
For a given layer l, with inputs z_i^l and outputs z_k^{l+1}

$$\delta_i^l = \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_i^l}$$

Similarly, for layer l that depends upon parameters θ^l,

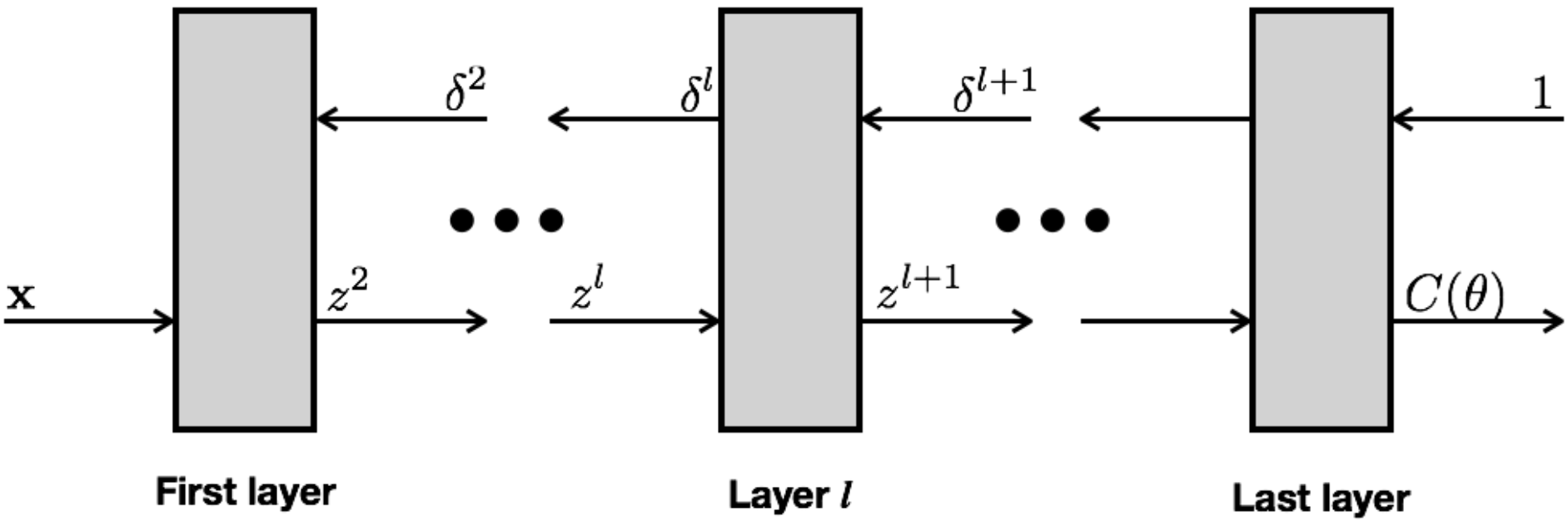
$$\frac{\partial C(\theta)}{\partial \theta^l} = \sum_k \frac{\partial C(\theta)}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial \theta^l} = \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial \theta^l}$$

In our 2-class softmax classifier only layer 1 has parameters (θ_0 and θ_1).



Layered architectures

As long as we have differentiable layers, i.e., we can compute $\frac{\partial z_k^{l+1}}{\partial z_i^l}$, we can use *backpropagation* to update the parameters θ to minimize the cost $C(\theta)$.



Backpropagation

- Set z^1 equal to input $\mathbf{x}$.
- Forward pass: compute $z^2, z^3, \ldots$ layers $1, 2, \ldots$ activations.
- Set δ at the last layer equal to 1
- Backward pass: backpropagate $deltas$ all the way to first layer.
- Update θ
- Repeat

Summary

- Layered view of logistic regression and softmax
- Backpropogation